VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2021 David Šidlo

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV BIOMEDICÍNSKÉHO INŽENÝRSTVÍ
DEPARTMENT OF BIOMEDICAL ENGINEERING

UMĚLÁ INTELIGENCE PRO PREDIKOVÁNÍ SEPSÍ Z KLINICKÝCH SIGNÁLŮ
ARTIFICIAL INTELLIGENCE FOR PREDICTING SEPSIS FROM CLINICAL SIGNALS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **David Šidlo**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Tomáš Vičar**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Biomedicínská technika a bioinformatika**

Ústav biomedicínského inženýrství

Student: David Šidlo ***ID:*** 203209
Ročník: 3 ***Akademický rok:*** 2020/21

NÁZEV TÉMATU:

Umělá inteligence pro predikování sepsí z klinických signálů

POKYNY PRO VYPRACOVÁNÍ:

1) Prostřednictvím odborné literatury se seznamte s metodami umělé inteligence hlubokého učení pro zpracování signálů. 2) Proveďte literární rešerši se zaměřením na rekurentní neuronové sítě. 3) Na základě literární rešerše vyberte nejvhodnější metodu pro predikci sepse a ve zvoleném programovacím prostředí ji otestujte. 4) Metodu realizujte pro dostupná data. 5) Úspěšnost otestujte a statisticky vyhodnoťte pomocí vhodných metrik.

DOPORUČENÁ LITERATURA:

[1] HOCHREITER, Sepp a Jürgen SCHMIDHUBER. Long Short-Term Memory. Neural Computation. 1997, 9(8), 1735-1780. DOI: 10.1162/neco.1997.9.8.1735. ISSN 0899-7667.

[2] GRAVES, Alex. Supervised sequence labelling with recurrent neural networks. New York: Springer, c2012. Studies in computational intelligence, v. 385. ISBN 978-364-2247-965.

Termín zadání: 8.2.2021 ***Termín odevzdání:*** 28.5.2021

Vedoucí práce: Ing. Tomáš Vičar

doc. Ing. Jana Kolářová, Ph.D.
předseda rady studijního programu

ABSTRAKT

Tato bakalářská práce řeší otázku predikování sepsí z klinických signálů pomocí metod umělé inteligence. V teoretické části je vypracována literární rešerše na zakladní principy a fungování různých metod umělé inteligence. Větší důraz byl kladen na rekurentní neuronové sítě. Cílem praktické části byla implementace vhodné metody ve zvoleném programovacím prostředí. Jako vhodná metoda byla vybrána LSTM síť a časová konvoluční síť TCN. Nejlepších výsledků normalizované hodnoty utility score dosáhla TCN, a to 0,377 a sedmi vrstvá LSTM 0,356.

KLÍČOVÁ SLOVA

sepse, neuronové sítě, neuron, LSTM, umělá inteligence

ABSTRACT

This bachelor thesis deals with the issue of predicting sepsis from clinical data using artificial intelligence methods. In the theoretical part, a literature research is made on the basic principles and functioning of various methods of artificial intelligence. Greater emphasis was placed on recurrent neural networks. The aim of the practical part was to implement a suitable method in the chosen programming environment. The LSTM network and the temporal convolutional network TCN were chosen as suitable methods. The best results of the normalized value of the utility score were achieved by TCN, namely 0.377 and seven-layer LSTM 0.356.

KEYWORDS

sepsis, neural networks, neuron, LSTM, artificial intelligence

ŠIDLO, David. *Umělá inteligence pro predikování sepsí z klinických signálů*. Brno, 2021, 56 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Tomáš Vičar

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Umělá inteligence pro predikování sepsí z klinických signálů“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Tomášovi Vičarovi za odborné vedení, vstřícnost, trpělivost, konzultace a podnětné návrhy k práci. Dále bych chtěl poděkovat rodině, přítelkyni a všem kamarádům, kteří mě v práci podporovali.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Jednou z hlavních příčin smrti důsledku infekce je sepse, která každý rok postihne okolo třiceti milionů pacientů, z nich 30% zemře. Mortalita sepse je již vyšší než u infarktu myokardu. V poslední letech se o ní hovoří jako o nejčasnější příčině smrti vůbec. K včasné detekci sepse pomáhá umělá inteligence.

Umělá inteligence, konkrétně neuronové sítě, jsou jakousi analogií biologického mozku. Pokroky v oblasti umělé inteligenci neustále rostou a používání umělé inteligence v medicíně má čím dál tím větší význam. Velkou roli hraje v diagnostice, klinickém rozhodování a personalizované medicíně.

V této práci je řešena otázka včasné detekce sepse z poskytnutých dat pomocí vhodných metod umělé inteligence. První část literární rešerše obsahuje základní principy neuronových sítí, jakožto jeden z modelů umělé inteligence. Je zde popsána historie neuronových sítí, analogie k biologickému neuronu, důležitost aktivační funkce, jednotlivé vrstvy, vysvětlen je i způsob učení neuronových sítí a příklady různých optimalizačních algoritmů. Druhá část práce se zabývá fungováním hlubokých neuronových sítí, kde největší zaměření bylo kladeno především na sítě rekurentní. Třetí část je věnována obecné charakteristice a vysvětlení pojmu sepse, společně s metodami, které se používají doposud na odhalení a predikování sepse. Na základě zpracování literární rešerše byla zhotovena čtvrtá část práce, která popisuje dataset, implementaci sítě long short term memory (LSTM) a časové konvoluční sítě (TCN). Poslední část zahrnuje vyhodnocení a srovnání dosažených výsledků.

1 Neuronové sítě

1.1 Počátky neuronových sítí

První model neuronu, o který se zasloužili Warren McCulloch a Walter Pittse, byl vytvořen roku 1943. Pravidlo učení (Hebovské učení) pro samotný neuron bylo navrženo a shrnuto v knize Organization of behavior od Donalda Hebba roku 1949. Ostré vystupování, zejména mužů Martina Mínského a Seymoura Parpeta, mělo za následek vydaní článku Perceptron roku 1969, který popisoval nedokonalost neuronu naučit se exkluzivní disjunkci (XOR). To mělo za následek období nezájmu o tuto oblast a v dalších letech nedošlo k významným objevům.

Důležitým milníkem byla publikace z roku 1986 o algoritmu BackPropagation (využití u vrstevnatých neuronových sítí). Autorem byl fyzik John Hopfield. Úspěchem v této době bylo zejména vytvoření systému NETtalk, který dokázal konverzi psaného textu do hovorové angličtiny, a tím nahradil starší DECtalk. Roku 1986 přišla novinka ve formě vícevrstvých neuronových sítí, ovšem vyskytl se problém, jak rozšířit Widrow-Hoffovo pravidlo na více vrstev. Tento problém se snažili vyřešit vědci. Nejznámějším z nich byl David Rumelhart (člen psychologického oddělení na Stanfordské univerzitě), který přišel s podobnou myšlenkou, jakou využívají dnešní zpětné propagační sítě. Ve výsledku se zjistilo, že sítě Backpropagation (zpětné šíření chyby) jsou kvůli mnoha iteracím pomalejší na učení. Říkáme jim tzv. „slow learners“.

Roku 1997 byla představena rekurentní neuronová sít Long short-term memory (LSTM) Seppem Hochreiterem. Dodnes je LSTM často používána. V dnešní době je známé, že algoritmy, které se používaly v 80. letech, fungují správně. Dříve ovšem nebyla tak vyspělá technika, a tím pádem byly tyto algoritmy výpočetně náročné pro hardware, se kterým se operovalo. Dnes je kladen důraz na algoritmy učení, které se používají zejména pro sady obsahující velké množství dat. [1] [2]

1.2 Umělý neuron

Neuron neboli nervová buňka, je základní stavební a funkční jednotka celé nervové tkáně. Jedná se o velmi specializované nervové buňky. Neurony mohou přijímat, zpracovávat a přenášet signály jak z vnitřního, tak i z vnějšího prostředí. Odpovědí na podněty signálů jsou reakce organismu. Neuron je složen z těla (soma), axony (dlouhé výběžky), dendrity (krátké výběžky). Dendrity jsou dostředivé části, jejichž funkce je příjem vstupní informace (vzruch z prostředí). Naopak Axony, které jsou odstředivé, vedou informaci v podobě nervového vzruchu z těla neuronu.

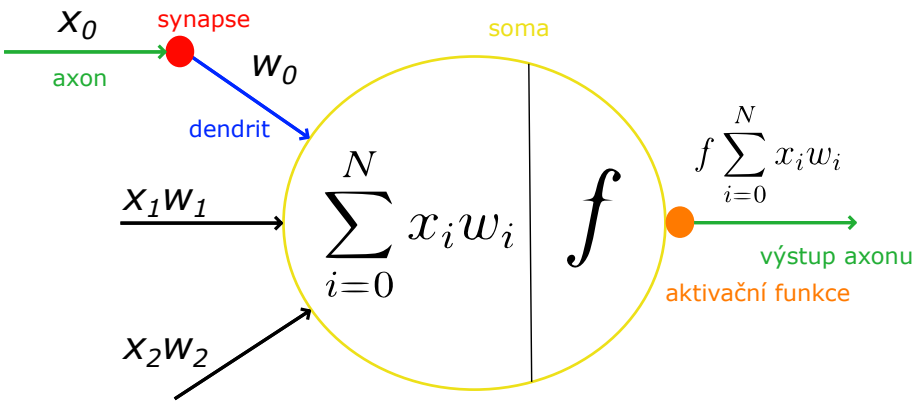
Model umělého neuronu je spjatý s modelem biologického neuronu. Jedná se o základní výpočetní jednotku ve všech neuronových sítí. Oba dva mají podobnou funkci. Model umělého neuronu je znázorněn na Obr. 1.1. U neuronu počítáme s více vstupy n a pouze jedním výstupem. Vstupy nám pak tvoří vstupní vektor $\boldsymbol{x} = (x_1, x_2..., x_N)$. Dále musíme brát v potaz, že každý z těchto vstupů bude měnit vnitřní potenciál určitou váhou $\boldsymbol{w} = (w_1, w_2, ..., w_N)$. Další možný vstup je prahová hodnota θ. Suma těchto informací ze vstupů společně s jejich váhami nám vyvolá aktivační funkci $f(x)$. Vstupy, které jsou upraveny váhami, se společně s prahem sčítají. Výsledek tohoto součtu nazýváme vnitřní potenciál neuronu. Funkce umělého neuronu je popsána rovnicí:

$$y = f \sum_{i=1}^{N} \boldsymbol{x}_i \boldsymbol{w}_i + \theta, \tag{1.1}$$

kde $\boldsymbol{x}_i$ jsou vstupy do neuronu, y je výstup z neuronu, $\boldsymbol{w}_i$ jsou určité váhy vstupů, θ je práh a f je aktivační funkce. [2] [3] V praxi se často setkáváme se zjednodušenou rovnicí:

$$y = f \sum_{i=0}^{N} \boldsymbol{x}_i \boldsymbol{w}_i, \tag{1.2}$$

kde $\boldsymbol{x}_0$ je nultý vstup, který se nastavuje na hodnotu 1 a $\boldsymbol{w}_0$ je nultá váha, která odpovídá prahu θ. Výsledkem je lineární kombinace.



Obr. 1.1: Model matematického neuronu přirovnaný k biologickému neuronu.

Můžeme si představit, že vstupy umělého neuronu jsou vektory $\boldsymbol{x}_i$, ty odpovídají dendritům neuronu biologického. Váhy $\boldsymbol{w}_i$ nám odpovídají synapsím biologického neuronu. Do těla neuronu pak patří celá suma a aktivační funkce, výstup y nám odpovídá axonu. Přirovnání matematického neuronu k biologickému prezentuje Obr 1.1

Tab. 1.1: Přehled nejpoužívanějších aktivačních funkcí.

Aktivační funkce	**Matematická rovnice**	**Obor hodnot**
Sigmoidální	$f(x) = \frac{1}{1-e^{-x}}$	$(0, 1)$
Tanh	$f(x) = tanh(x)$	$(-1, 1)$
Gaussova	$f(x) = e^{-x^2}$	$(0, 1\rangle$
Skoková	$f(x) = sign(x)$	$\{-1, 1\}$
ReLu	$f(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases}$	$\langle 0, \infty)$
Leaky ReLu	$f(x) = \begin{cases} x, & x \geq 0 \\ 0.01x, & x < 0 \end{cases}$	$(-\infty, \infty)$

1.3 Aktivační funkce

Jedná se o přenosovou funkci. Na základě toho, jakou aktivační funkci použijeme, dostáváme odpovídající výstup neuronu $y = f(x)$. Mezi aktivační funkce patří například funkce: sigmoidální, skoková, tanh, gaussova, ReLu, LeakyReLu, ELU. Přehled často používaných funkcí a jejich grafické zobrazení představuje Tab. 1.1 a Obr. 1.2 [3]

Sigmoidální

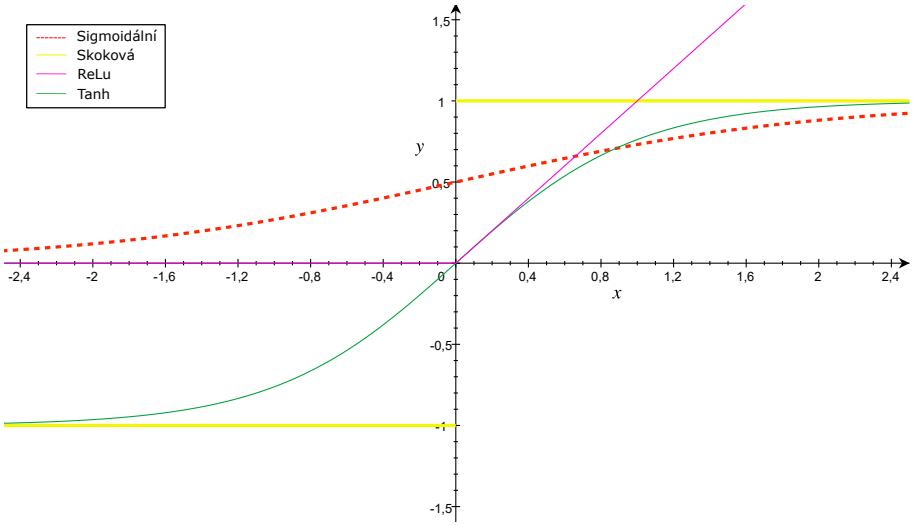
Tato funkce existuje na intervalu $(0, 1)$, proto se používá pro modely, u kterých musíme předpovědět očekávaný výstup. Sigmoidální funkce je diferencovatelná, což znamená, že dokážeme najít sklon křivky.

Tanh

Aktivační funkce je podobná sigmoidální. Rozsah této funkce je $(-1, 1)$. Vstupy se zápornou hodnotou se nám budou zobrazovat ve tvaru záporném, zatímco nulové vstupy se zobrazí blízko nuly. Stejně jako sigmoidální, je i tato funkce diferencovatelná.

ReLu

Tato aktivační funkce přenese kladné vstupní hodnoty na výstup, pro ostatní hodnoty je výstup nulový. To odpovídá rozsahu funkce od $\langle 0, \infty)$. Jedná se o nejpoužívanější aktivační funkci současnosti. [3]



Obr. 1.2: Grafy velmi často používaných aktivačních funkcí. Jedná se o grafy funkce sigmoidální, skokové, ReLu a tanh.

1.4 Vrstvy neuronových sítí

Každá neuronová síť může být seskládána s více vrstev. [4] Zobrazení jednotlivých vrstev prezentuje Obr. 1.3. Mezi základní typy patří:

1. Vstupní vrstva - má za úkol příjem vstupních dat a předávání těchto dat do skrytých vrstev. Vstupní vrstva je na počátku celého pracovního postupu umělé neuronové sítě.

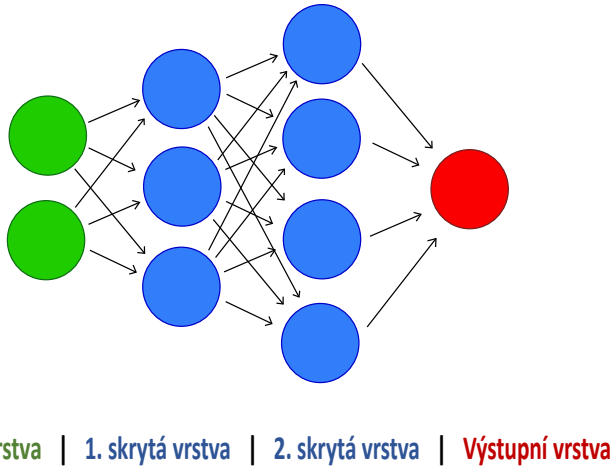
2. Skryté vrstvy - jedná se o vrstvy neuronů, které jsou přidávány mezi vrstvu vstupní a výstupní. V těchto vrstvách získávají neurony potřebné informace pomocí vhodného procesu. Počet skrytých vrstev souvisí s druhem vybrané topologie.

3. Výstupní vrstva - odpovídá za vytvoření výsledného výstupu dané sítě. Výstup nám závisí na procesech, které se odehrávají v předchozích skrytých vrstvách. Jde o poslední vrstvu v umělé neuronové síti.

SoftMax vrstva

Tato vrstva bývá nejčastěji výstupní vrstvou v neuronové síti. Výstupní hodnoty nabývají rozsahu $<0, 1>$. U SoftMax dochází k přeměně vstupních hodnot vektoru x na hodnoty výstupního vektoru y, po sečtení všech hodnot výstupního vektoru y dostáváme číslo rovno jedné. SoftMax se používá nejčastěji pro kvalifikační úlohy. [5] Je popsána rovnicí:

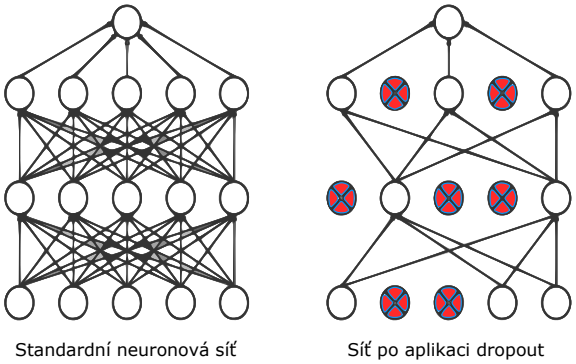
$$y_i = \frac{e^{x_j}}{\sum_{i=1}^{N} e^{x_k}} \tag{1.3}$$



Obr. 1.3: Zobrazení a popis jednotlivých neuronových vrstev

Dropout vrstva

Vrstva dropout pomáhá neuronové síti k tomu, aby nedošlo k její přetrénovanosti (overfitting). Používá se jen v období tréninku neuronové sítě. Při použití na testovací sadu by došlo k zobrazení náhodných výstupů. Pracuje tak, že při každé iteraci vybere náhodnou množinu spojů neuronu a tuto množinu deaktivuje. To má za následek menší propojení mezi neurony (menší závislost). Dropout vrstva bývá nejčastěji umístěna za aktivační funkci. Důležité je zmínit tzv. Drop faktor. Jedná se o hyperparametr p, který se vyskytuje v hodnotách na intervalu $(0, 1)$. Tento hyperparametr nám určuje pravděpodobnost deaktivace neuronu. Nejčastěji je dropout – výpadek neuronů nastavován na 0,5, což deaktivuje polovinu neuronů v dané síti. Je dokázáno, že sítě s využitím dropoutové vrstvy dosahují lepších výsledků. [6] Fungování dropout vrstvy popisuje Obr. 1.4.



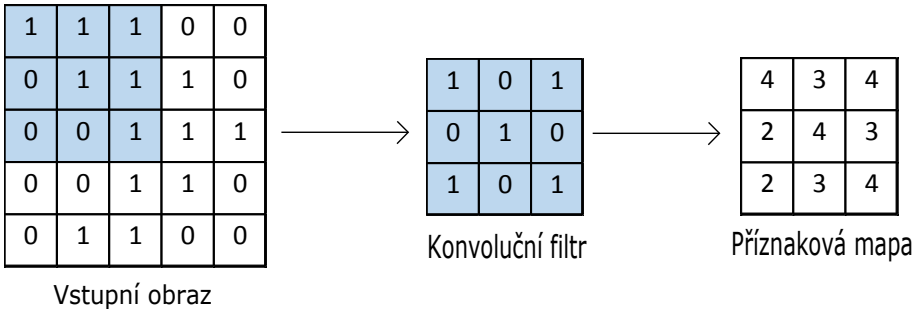
Obr. 1.4: Schéma na levé straně popisuje standardní chování neuronové sítě. V pravé části vidíme fungování sítě po aplikaci vrstvy dropout.

Vrstva konvoluční

Tato vrstva je složena z velmi malých filtrů, které jsou však schopny zaujmout celý objem vstupních dat a z těch se následně učit. Tyto filtry pracují tak, že vstupní obrázek projíždí z jedné strany na druhou (šířka) a následně zespodu směrem nahoru (výška), a tím získává důležité příznaky daného objektu tzv. příznakovou mapu. Často se používá více filtrů na jeden obrázek s různými váhami. Výsledný počet příznakových map závisí na počtu použitých filtrů. Fungování konvoluční vrstvy prezentuje Obr. 1.5. Konvoluční vrstva je popsána rovnicí:

$$G[m,n] = (f * h)[m,n] = \sum_j \sum_k h[j,k] f[m-j, n-k], \tag{1.4}$$

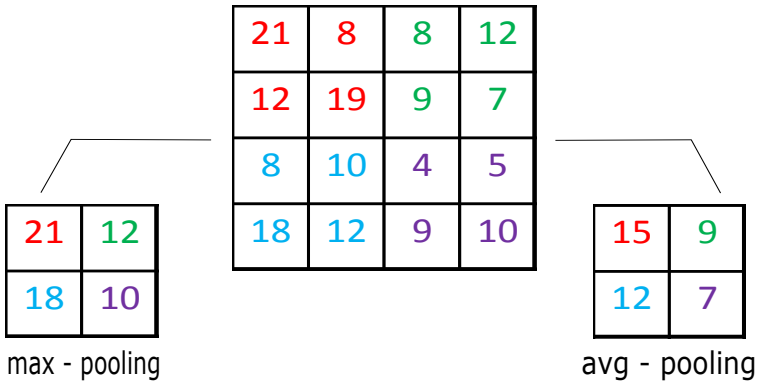
kde f je vstupní obraz a h je filtr (kernel), m reprezentují řádky a n sloupce. [7]



Obr. 1.5: Schéma popisuje fungování principu konvoluce s použitím konvolučního filtru (kernel).

Pooling vrstva

Účelem je sdružení hodnot pomocí redukce do jedné hodnoty. Pooling vrstva je hojně využívána v architekturách konvolučních sítí. Jedná se o maximální sdružení a podvzorkování. Vrstva pooling se dá rozdělit podle toho, jaké rozdělení dat používáme na max–pooling, sum–pooling, avg–pooling atd. Konkrétně pro max–pooling nebo hledání maxima se vybere v našem případě z oblasti 2x2 nejvyšší číslo. Následně se napíše na výstup. U avg–poolingu dojde k sečtení hodnot dané příznakové mapy a následně vydělením počtem hodnot. Obecně pooling vrstva slouží k redukci parametrů, zvýšení výkonu sítě a přetrénovanosti. Jako příklad je uveden max–pooling a avg–pooling na Obr. 1.6 [8]



Obr. 1.6: Fungování vrstvy pooling, levá strana popisuje výpočet max - poolingu, pravá strana výpočet avg - poolingu.

1.5 Učení neuronových sítí

Učení neuronových sítí je dále rozděleno na dvě základní kategorie. [4]

1) Učení s učitelem

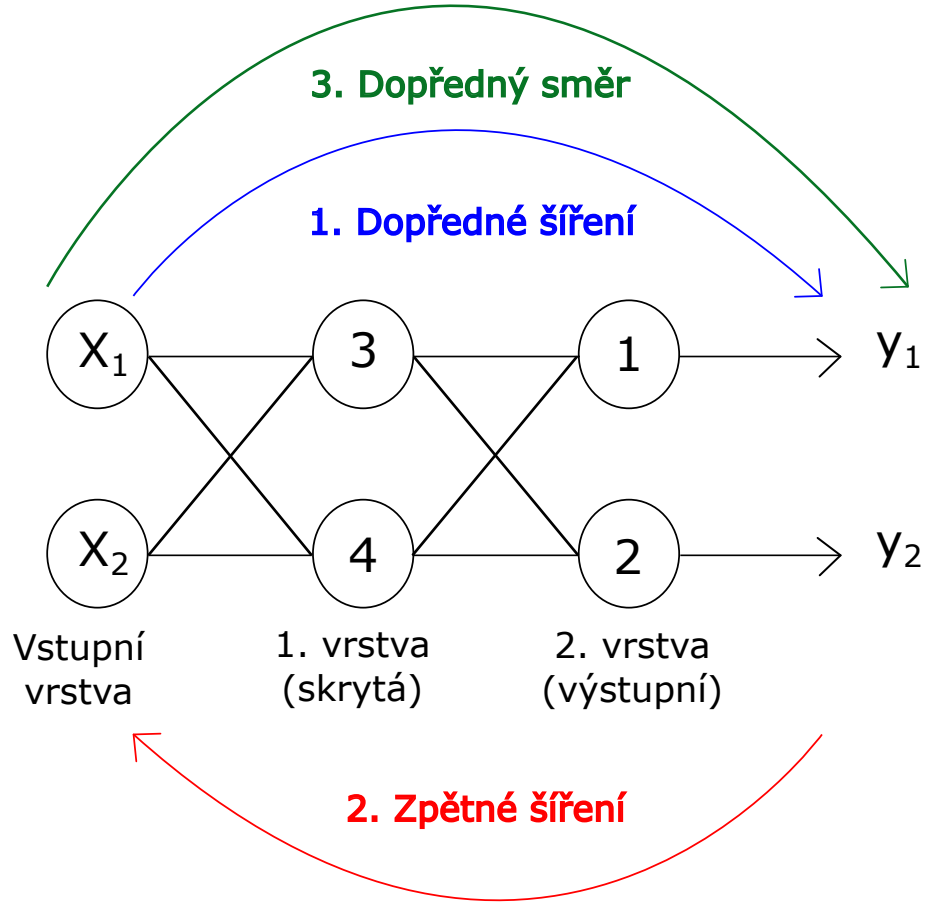
Jedná se o systém, který je založen na metodě strojového učení. Pro učení používá dva parametry – podnět a jeho následnou odezvu. Mezi vstupem (podnět) a výstupem (odezva) se může vyskytnout jistá závislost, která nám je ovšem neznámá. Známa je pouze konečná množina dvojic (podnět a odezva). Proto je nutné znovu sestavení modelu vztahu mezi podnětem a jeho odezvou. Po sestavení algoritmu, který reaguje na libovolný objekt správnou odpovědí, můžeme učení považovat za zdařilé. Systém je sestaven z použitých dat, soustavou podnětů, které jsou dodávány z vnějšího prostředí a regulátoru vnitřních parametrů. Jako regulátor se dá použít člověk (učitel). Regulátor je totiž schopen reagovat na základě předem definovaných pravidel na podněty vnějšího prostředí. Pravidla následně pozměňují stav paměti systému. Učení s učitelem patří mezi nejčastější proces, který se používá při učení umělé neuronové sítě.

2) Učení bez učitele

Můžeme o něm hovořit jako o určitém samostudiu. Jde o systém, který je založen na metodě strojového učení. Dochází k učení systému, bez nutného zasahování člověkem. Tato metoda je vhodná k použití jen v případech, kdy známe popis množiny objektů a potřebujeme najít nějaké propojení (závislost, vztah), které je mezi objekty.

Algoritmus zpětného šíření chyby

Algoritmus zpětného šíření chyby (Backpropagation) se nejčastěji používá pro učení vícevrstvých neuronových sítí. Využívá se v procesu učení s učitelem. Algoritmus používá tzv. gradientní metodu. Na základě směru gradientu udává změnu chybovosti v neuronové síti spojenou se změnou vah pro jednotlivé synapse. Hlavním principem tohoto algoritmu je poupravení vah tak, aby se snížila hodnota chybové funkce. Zjednodušeně tento algoritmus můžeme provést pomocí násobení matic. Algoritmus se dělí na základní tři části: dopředné šíření, zpětné šíření chyby a aktualizace (adaptace) vah neuronů. [9] Schéma popisující algoritmus zpětného šíření chyby je na Obr. 1.7.



Obr. 1.7: Schéma popisující algoritmus zpětného šíření chyby. Skládá se ze tří částí: 1. Dopředné šíření - výpočet výstupu sítě a odchylky, 2. Zpětné šíření - výpočet odchylek na všech neuronech, 3. Dopředný směr: úprava vah (adaptace) všech neuronů.

Gradientní sestup

Z angličtiny Gradient Descent, jedná se o základní optimalizační algoritmus, který se využívá při učení neuronových sítí. Spočívá v iterační úpravě parametrů v závislosti na gradientu chybové funkce. Aby byla provedena úprava parametru v dané iteraci, musí gradientní sestup projít všechny vzorky v trénovací sadě. Rovnice pro úpravu vah je popsána:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma \cdot \bigtriangledown L(\boldsymbol{w}_t) \tag{1.5}$$

Ve vzorci je koeficient ovlivňující rychlost učení γ, gradient chybové funkce $\bigtriangledown L(w_t)$. Tento algoritmus se používá u zpětného šíření chyby (Backpropagation), kde potřebujeme poupravit váhy v neuronové síti. Nevýhoda algoritmu: velmi pomalý, náročný na výpočet. [10]

Další metody optimalizačních algoritmů jsou uvedeny níže. Jde o modifikace gradientního sestupu.

Stochastický gradientní sestup

Z angličtiny Stochastic Gradient Descent, oproti gradientnímu sestupu je rychlejší a méně náročnější na výpočet. K uvedenému výpočtu se používá pouze náhodně vybraný vzorek z trénovací sady. Úprava vah v konkrétní iteraci se uskutečňuje při počítání s jedním vzorkem z trénovací sady. [10] Výpočet aktualizace vah je popsán právě pro jeden vzorek z trénovací sady, rovnicí:

$$\boldsymbol{w}_{t+1,i} = \boldsymbol{w}_{t,i} - \gamma \cdot \bigtriangledown L(\boldsymbol{w}_{t,i}, \boldsymbol{x}_i, y_i), \tag{1.6}$$

kde $\boldsymbol{x}_i$ je aktuální vstup a y_i je požadovaný výstup.

Mini - Batch Gradient Descent

Jedná se o jakýsi kompromis mezi gradientním sestupem a stochastickým gradientním sestupem. U Mini - Batch Gradient Descent (MB-GD) dochází k úpravě parametrů modelu pomocí menších skupin vzorků v trénovací sadě. Počítáme z $1 < k < n$, kde n je počet vzorků v celé trénovací sadě, k je počet vybraných vzorků (menší skupina). U MB-GD dochází k rychlejší konvergenci, než u gradientního sestupu (častější aktualizace vah). Oproti gradientnímu sestupu dochází k rychlejšímu výpočtu. Naopak výpočetně pomalejší je v případě porovnání se stochastickým gradientním sestupem, a to kvůli většímu množství vektorových operací. MB-GD nepočítá pouze s jedním vzorkem, jako v případě stochastického gradientního sestupu, ale s menší skupinou. MB-GD je velmi používaný a efektivní. [11]

Momentum

Je metoda, která pomáhá urychlit vektory gradientů ve správném směru, což vede k rychlejší konvergenci. [12] Momentum je popsáno rovnicí:

$$\boldsymbol{v}_t = \beta \cdot \boldsymbol{v}_{t-1} + (1 - \beta) \cdot \bigtriangledown L(\boldsymbol{w}_t), \tag{1.7}$$

kde β je konstanta momenta, která se nejčastěji nastavuje na hodnotu $\beta = 0,9$. Aktualizace vah je popsána rovnicí:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \gamma \boldsymbol{v}_t, \tag{1.8}$$

kde γ je koeficient ovlivňující rychlost učení.

Momentum nám pomáhá v menší oscilaci a rychlejší konvergenci u chybové funkce.

Adaptive Moment Estimation (ADAM)

Jedná se metodu, na kterou se dá pohlížet jako na spojení RMSProp a AdaGrad. ADAM uplatňuje i funkčnost momenta. Je založen na výpočtu adaptivní rychlosti učení, což znamená, že vypočítává individuální rychlost učení pro různé parametry. [11][16] Aktualizuje parametry pomocí rovnic:

$$\boldsymbol{m}_t = \beta_1 \boldsymbol{m}_{t-1} + (1 - \beta_1) \bigtriangledown L(\boldsymbol{w}_t) \tag{1.9}$$

$$\boldsymbol{v}_t = \beta_2 \boldsymbol{v}_{t-1} + (1 - \beta_2) \bigtriangledown L(\boldsymbol{w}_t)^2 \tag{1.10}$$

kde β_1 a β_2 jsou hyperparametry, které se nastavují na hodnoty $\beta_1 = 0,9$ a $\beta_2 = 0,999$. Předběžné odhady prvního a druhého momentu jsou popsány rovnicemi:

$$\hat{\boldsymbol{m}}_t = \frac{\boldsymbol{m}_t}{1 - \beta_1^t} \tag{1.11}$$

$$\hat{\boldsymbol{v}}_t = \frac{\boldsymbol{v}_t}{1 - \beta_2^t} \tag{1.12}$$

Předběžný odhad prvního momentu je $\hat{\boldsymbol{m}}_t$, pak odhad druhého momentu $\hat{\boldsymbol{v}}_t$. Výpočet pro úpravu parametrů má tvar:

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \frac{\gamma \cdot \hat{\boldsymbol{m}}_t}{\sqrt{\hat{\boldsymbol{v}}_t} + \varepsilon}, \tag{1.13}$$

kde $\varepsilon = 10^{-8}$ a γ je koeficient ovlivňující rychlost učení. ADAM patří k optimalizačním algoritmům, které se často používají. Má velice dobré výsledky ve srovnání s jiným optimalizačním algoritmem. Rychlost učení je velmi rychlá a efektivní.

Další algoritmy, které se používají na optimalizaci parametrů: Nesterov, AdaGrad a RMSProp. [11]

1.6 Chybová funkce

Součástí každé neuronové sítě je chybová funkce (Loss function). Chybová funkce nám říká, jak výrazně se liší výsledek od výsledku, který bychom požadovali. Chybových funkcí registrujeme mnoho. Nejznámější jsou: střední kvadratická odchylka (MSE), průměrná absolutní odchylka (MAE), vzájemná entropie, dice loss nebo hinge loss. [2]

Vzájemná entropie

Vzájemná entropie (Cross entropy) je funkce, jejíž ztráta se zvyšuje tím, jak se predikovaná hodnota vzdaluje od skutečné hodnoty. Využívá se nejčastěji pro klasifikační úlohy. Nabývá hodnot od 0 do 1. Dokonalý model by měl obsahovat ztrátovou funkci rovno 0. Vzájemná entropie je popsána rovnicí:

$$H(y,p) = -\sum_{i=1}^{M} y_i \log(p_i), \tag{1.14}$$

kde y je skutečná hodnota výstupu, p je pravděpodobnost výstupu.

Kdybychom chtěli popsat počet dvou tříd $M = 2$, můžeme použít rovnici:

$$H(y,p) = -(y \log(p) + (1-y) \log(1-p)) \tag{1.15}$$

Střední kvadratická odchylka

Z angličtiny Mean Squared Error (MSE), je jedna ze základních regresních chybových funkcí. MSE sčítá hodnoty výstupu, předpokládaného výstupu a konečný výsledek umocní druhou mocninou. Tato funkce je často označovaná jako L2 ztráta. [14] Nabývá hodnot od $(0, \infty)$. Je popsána rovnicí:

$$MSE = \frac{\sum_{i=1}^{M} (y_i + y_i^p)^2}{M}, \tag{1.16}$$

kde y_i je skutečná hodnota výstupu, y_i^p je predikovaná hodnota výstupu.

Průměrná absolutní odchylka

Z angličtiny Mean Absolute Error (MAE), je jedna z dalších uvedených chybových funkcí. Opět je používána pro regresní modely. Jedná se o sečtení v absolutní hodnotě rozdílů mezi výstupem a předpokládaným výstupem. Tato funkce je často označována jako L1 ztráta. [14] Nabývá hodnot od $(0, \infty)$. Je popsána rovnicí:

$$MAE = \frac{\sum_{i=1}^{M} |y_i + y_i^p|}{M}, \tag{1.17}$$

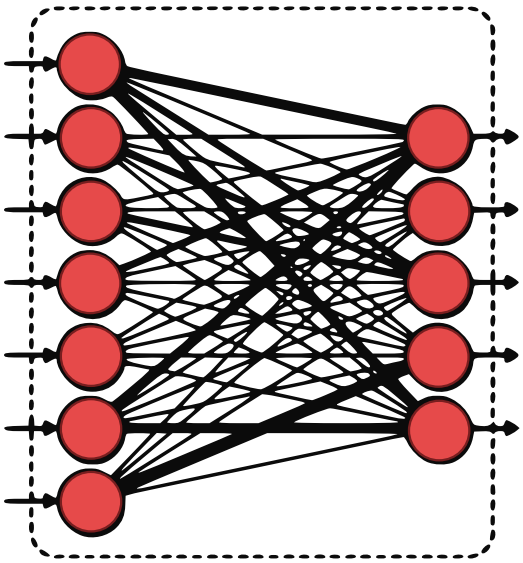
kde y_i je skutečná hodnota výstupu, y_i^p je predikovaná hodnota výstupu.

2 Hluboké neuronové sítě

Hluboké učení neboli Deep Learning je podmnožinou strojového učení, které se zabývá algoritmy inspirovanými strukturou a funkcí mozku (umělé neuronové sítě). Jde o jakousi posloupnost konceptů, která umožňuje počítači naučit se složité operace. Kdybychom chtěli ukázat, jak jsou tyto koncepty propojeny, sestrojili bychom graf, který by obsahoval mnoho vrstev – z toho název Hluboké učení. V praxi dochází k testovaní takových modelů, které obsahují velké množství vstupních dat. Každý rok dochází v této oblasti k velkolepým výsledkům v odvětví zpracování (rozpoznávání) řeči, zvuků, obrazů a jiných dat. [2]

2.1 Plně propojené sítě

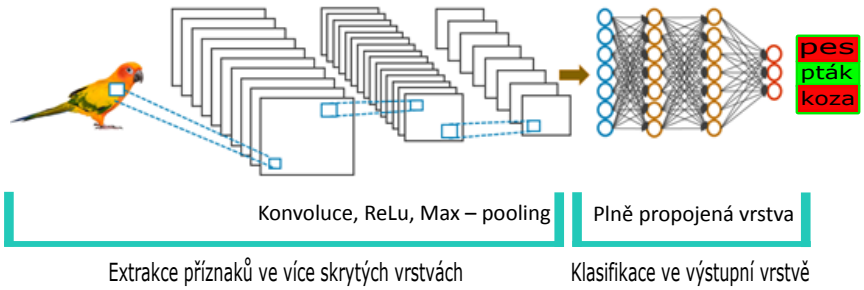
Z angličtiny Fully Connected Network, jsou sítě složené z neuronů, které jsou propojeny s vektorem vstupních dat. Takže každý vstup je spojen s každým výstupem. Jako výstupní vektor pak dostáváme vektor o stejné velikosti, jako je počet neuronů v síti. Spojení mezi vstupním vektorem a neurony je ovlivňováno určitými váhami. Plně propojená vrstva se často používá u konstrukce konvolučních sítí, kde je na pozici konečné vrstvy. Plně propojené vrstvy dokáží aproximovat různé spojité funkce. Plně propojené vrstvy můžeme vidět na Obr. 2.1. [2]



Obr. 2.1: Ukázka fungování plně propojené sítě, kde všechny neurony jedné vrstvy jsou propojeny s vrstvou následující.

2.2 Konvoluční neuronové sítě

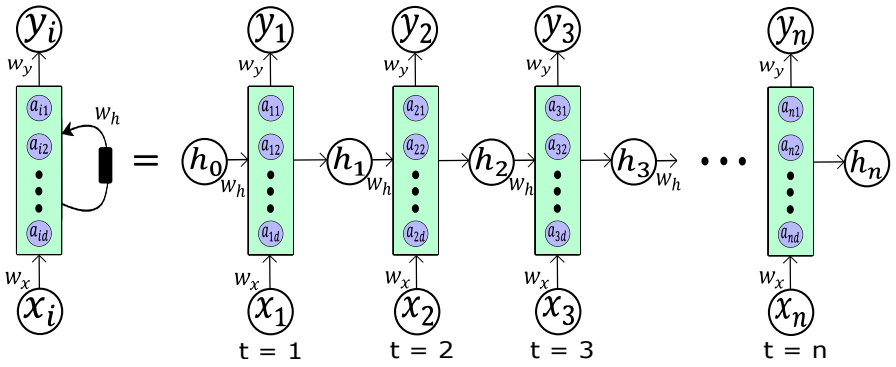
Z angličtiny Convolutional Neural Networks (CNN). Konvoluční neuronové sítě, jsou dopředné neuronové sítě, které obsahují velké množství vrstev. Jsou obdobné jako obvyklé neuronové sítě. Základem je matematický model neuronu s měnícími se váhami a prahem. Dále tyto sítě obsahují diferencovatelnou funkci (ReLu) a výpočet chybové funkce. Nejčastěji se CNN používají při zpracování obrazů. Neurony CNN jsou proto uspořádány do třech rozměru, a to výšky, šířky a hloubky. Tímto se oddělují od klasických neuronových sítí. Konvoluční neuronová síť je složena s více podvrstev – vstupní konvoluční vrstva, vrstva pooling a plně propojené vrstvy. Podle toho, jaké používáme typy a struktury dané sítě, se může její složení měnit. Každá z těchto vrstev má určitou funkci. Jestliže chceme identifikovat dané objekty (obrázky), síť pracuje následovně. Vstup je ve formě obrázku, v první vrstvě dojde k rozpoznání pixelů podle barvy (tmavá, světlá), v další vrstvě by byly určeny hrany, v další tvary, a tak by to pokračovalo, dokud bychom na výstupu nedostali požadovaný obrázek. Architekturu konvoluční neuronové sítě popisuje Obr. 2.2. [2]



Obr. 2.2: Schéma fungování konvoluční sítě, naznačeny jsou i její základní vrstvy.

2.3 Rekurentní neuronové sítě

Z angličtiny Recurrent Neural Neworks (RNNs), jsou sítě, které patří do neuronových sítí pro zpracování dat. Jedná se o tzv. opakující se sítě, které se specializují spíše na zpracování sekvence hodnot $x_1, x_2, \ldots, x_n$ nebo vektorů $\boldsymbol{x} = (x_1, x_2..., x_n)$. Rekurentní sítě se dají používat pro zpracování obrazů stejně jako konvoluční, ale jejich větší specializace je využita při zpracování signálů, kde využíváme sekvence velké délky. V rekurentních sítích při učení dochází ke sdílení parametrů v různých částech modelu. Hlavní princip RNNs spočívá v uchování paměti předchozích vstupů, která zůstává ve vnitřním prostředí sítě, a to ovlivňuje výstup ze sítě. Nejznámějšími RNNs jsou: Long short term memory (LSTM) a Gated Recurrent Unit (GRU). Schéma 2.3 popisuje základní princip fungování RNNs. [2]



Obr. 2.3: Základní princip fungování rekurentních neuronových sítí

Zelené bloky nám představují skryté vrstvy. V těchto blocích si můžeme všimnout modrých koleček s vektorem $\boldsymbol{a}$, který nám značí skryté jednotky. Zelený blok pracuje jako aktivační funkce $tanh$, která působí na každou skrytou jednotku (modré kolečka). Vektor $\boldsymbol{h}_t$ je výstupní vektor skryté vrstvy v určitém čase t. Jsou zde váhy $\boldsymbol{w}_x, \boldsymbol{w}_y, \boldsymbol{w}_h$, vstupy $\boldsymbol{x}_i$ a výstupy y_i, kde $i = [1, 2, \ldots, n]$.

Rekurentní výpočet

Každá RNNs může být sestavena různým způsobem. Jedná se o dopředné neuronové sítě. Většina RNNs používají tuto rovnici na výpočet skrytých vrstev:

$$\boldsymbol{h}_t = tanh(\boldsymbol{w}_h \boldsymbol{h}_{t-1} + \boldsymbol{w}_x \boldsymbol{x}_t) \tag{2.1}$$

Síť se obvykle učí používat $\boldsymbol{h}_t$ jako druh ztrátového souhrnu aspektů souvisejících s minulými sekvencemi vstupů až do t.

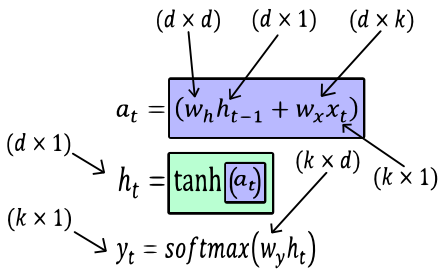
Predikce výstupu ze skryté vrstvy, mluvíme-li o klasifikaci, je popsána rovnicí:

$$y_t = softmax(\boldsymbol{w}_y \boldsymbol{h}_t) \tag{2.2}$$

Důležité je zmínit rozměry proměnných, které se používají v těchto výpočtech. Obr. 2.4

Trénování rekurentních neuronových sítí

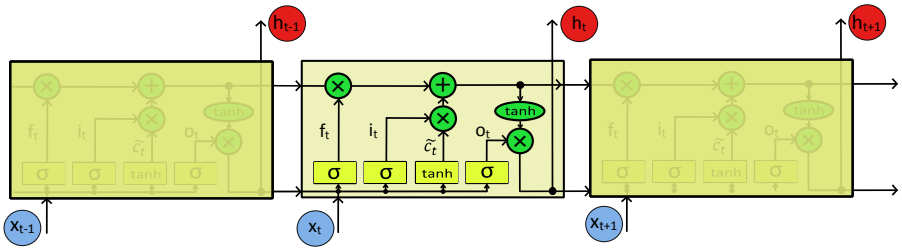
Rekuretní sítě se trénují na základě algoritmu zpětného šíření chyby (Backpropagation), přičemž potřebujeme síť dostat i do časové oblasti. Tento proces nazýváme algoritmus zpětného šíření v čase (Backpropagation through time). U tréninku RNNs mohou nastat potíže s váhami (problém mizejícího gradientu). Tyto potíže jsou řešeny sítěmi jako je síť Long Short term memory, nebo Gated Reccurent Unit. [2]



Obr. 2.4: Jedná se o popis, včetně rozměrů proměnných, jednotlivých používaných rovnic u práce s rekurentními neuronovými sítěmi.

2.3.1 Long short term memory (LSTM)

LSTM sítě jsou speciálním druhem patřící do skupiny zpětnovazebných neuronových sítí. LSTM sítě se dokáží naučit závislosti, ve kterých daný výstup závisí na dřívějším odlehlém vstupu. Sítě LSTM jsou nejčastěji používány pro rozpoznání řeči, skládání hudby, klasifikaci a rozpoznání textů. Jsou používány i pro predikování v oblasti časových řad. [15]



Obr. 2.5: Schéma zobrazuje jednotlivé skládání a fungování vnitřních stavů buněk, jakožto fungování základní jednotky LSTM.

Nejdůležitějším znakem LSTM je tzv. vnitřní stav buňky neboli cell state. Vnitřní stav buňky je podrobněji zobrazen na Obr. 2.5 a 2.6. Vnitřní stav ovlivňují pouze lineární interakce, a díky tomu nám dané informace prochází celým systémem bez ovlivnění. V LSTM jsou důležité tzv. brány (gates), které slouží k úpravě vnitřních stavů. Brány jsou složeny ze sigmoidální funkce a operace násobení. Sigmoidální funkce nám určuje čísla výstupu na intervalu $(0, 1)$, která slouží k procházení komponentů přes bránu. Pokud na výstupu registrujeme číslo 0, brána nebude propouštět nic. Naopak hodnota 1 zaručuje, že bude propuštěno vše.

Zapomínající brána

Z angličtiny forget gate layer. Zapomínající brána nám určuje, jaká informace bude z vnitřního stavu buňky odstraněna. Je popsána rovnicí:

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_f[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + b_f), \tag{2.3}$$

kde $\boldsymbol{W}_f$ je váha, b_f je práh a σ je sigmoidální funkce pro zapomínající bránu. Pracuje tak, že vezme výstup z předešlého modulu $\boldsymbol{h}_{t-1}$, současný vstup $\boldsymbol{x}_t$ a výsledkem je číslo v intervalu $(0, 1)$ pro každé číslo z vnitřního stavu buňky. Číslo na výstupu nám určuje, jak moc je nutné si zapamatovat informaci, která prošla bránou.

Vstupní brána a přizpůsobení současnému stavu

Rozhoduje o tom, jaké nové informace jsou doplněny do vnitřního stavu buňky. Tento děj primárně rozhodne, jaké hodnoty se budou přizpůsobovat současnému stavu. Výše uvedené skutečnosti ovlivňuje vstupní vrstva (input gate layer). Reakcí je aktualizace pomocí *tanh* vrstvy, která vytvoří vektor nových hodnot. Tento vektor bude zapojen do vnitřního stavu. Vstupní brána je popsána rovnicí:

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_i[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + b_i), \tag{2.4}$$

kde $\boldsymbol{W}_i$ je váha, b_i je práh, σ je sigmoidální funkce pro vstupní bránu, $\boldsymbol{h}_{t-1}$ je výstup z předešlého modulu a $\boldsymbol{x}_t$ je současný vstup.

Výpočet kandidátů na nový vnitřní stav je popsán rovnicí:

$$\tilde{\boldsymbol{c}}_t = tanh(\boldsymbol{W}_c[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + b_c), \tag{2.5}$$

kde $\boldsymbol{W}_c$ je váha, b_c je práh pro výpočet stavu buňky, $\boldsymbol{h}_{t-1}$ je výstup z předešlého modulu a $\boldsymbol{x}_t$ je současný vstup.

Výpočet nového vnitřního stavu je znázorněn rovnicí:

$$\boldsymbol{C}_t = \boldsymbol{f}_t * \boldsymbol{C}_{t-1} + \boldsymbol{i}_t * \tilde{\boldsymbol{c}}_t, \tag{2.6}$$

kde $\boldsymbol{C}_{t-1}$ jsou staré hodnoty vnitřního stavu, $\boldsymbol{C}_t$ jsou nové hodnoty a operace $*$ slouží k vektorovému násobení.

Výstupní brána

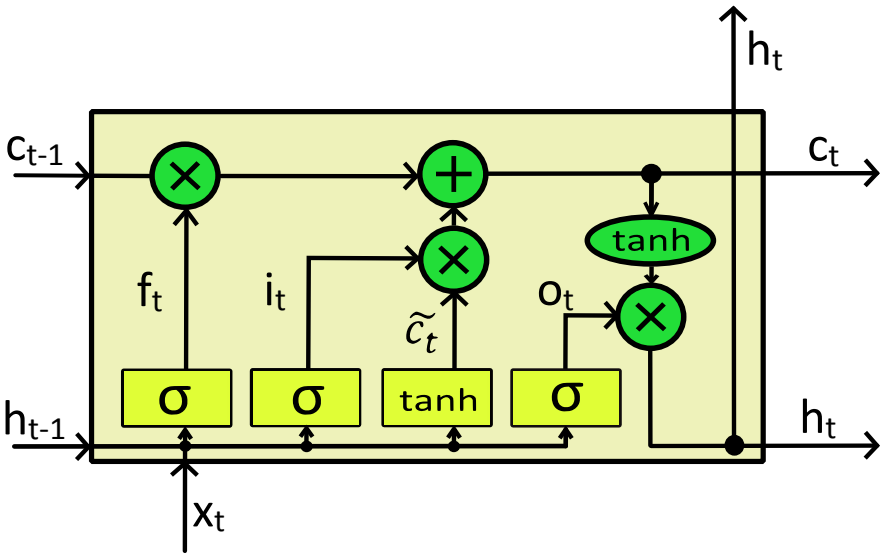
Poslední práci v sítích LSTM má na starost výstupní brána (output gate layer). Ta nám rozhoduje o tom, co bude na výstupu. Je závislá na aktuálním vnitřním stavu buňky a jejího vstupu pomocí výstupní brány. Vztah výstupní brány je popsán rovnicí:

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_o[\boldsymbol{h}_{t-1}, \boldsymbol{x}_t] + b_o), \tag{2.7}$$

kde $\boldsymbol{W}_o$ je váha, b_o je práh pro výstupní bránu, σ je sigmoidální funkce pro výstupní bránu, $\boldsymbol{h}_{t-1}$ je výstup z předešlého modulu a $\boldsymbol{x}_t$ je současný vstup.

Výstup je závislý na výpočtu součinu výstupní brány a výpočtu vnitřního stavu buňky. Výstup je popsán rovnicí:

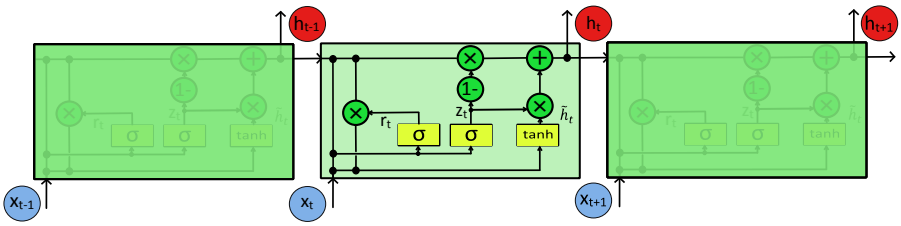
$$\boldsymbol{h}_t = \boldsymbol{o}_t * tanh(\boldsymbol{C}_t) \tag{2.8}$$



Obr. 2.6: Detailní zobrazení vnitřního stavu buňky sítě LSTM

2.3.2 Gated Recurrent Unit (GRU)

GRU je další speciální typ rekurentních neuronových sítí. Sítě GRU jsou velmi podobné dříve zmiňovaných LSTM. Skládají se ze dvou bran, a to z brány aktualizační a resetovací. Brány nám představují vektory, které rozhodují o tom, jaká informace bude předána na výstup. Důležitou vlastností těchto sítí je, že dokáží uschovávat informace z minulosti, aniž by časem došlo k jejich odstranění. Tato vlastnost je zdokonalována tréninkem sítě. Napojení a fungování buněk GRU je zobrazeno v 2.7 a detailnější popis vnitřního stavu buňky je zobrazen na Obr. 2.8. [16]



Obr. 2.7: Schéma zobrazuje jednotlivé skládání a fungování vnitřních stavů buněk, jakožto fungování základní jednotky GRU.

Aktualizační brána

Tato brána slouží k předání informacích z minulosti do budoucnosti modulu. Tato vlastnost je velice důležitá. Díky tomu dokáže brána vyloučit riziko mizejícího gradientu. Aktualizační brána pomáhá zachytit dlouhodobé závislosti v časové řadě. Je popsána rovnicí:

$$\boldsymbol{z}_t = \sigma(\boldsymbol{W}_z \boldsymbol{x}_t + \boldsymbol{U}_z \boldsymbol{h}_{t-1}), \tag{2.9}$$

kde $\boldsymbol{W}_z$ je váha současného vstupu $\boldsymbol{x}_t$, $\boldsymbol{U}_z$ je váha výstupu z předešlého modulu $\boldsymbol{h}_{t-1}$ a σ je sigmoidální funkce pro aktualizační bránu.

Resetující brána

Mohli bychom říci, že resetující brána vykonává opačnou práci než brána aktualizační. Resetující brána je používána pro minulé informace, které mají být zapomenuty. Pomáhá tedy zachytit krátkodobé závislosti v časové řadě. Je popsána rovnicí:

$$\boldsymbol{r}_t = \sigma(\boldsymbol{W}_r \boldsymbol{x}_t + \boldsymbol{U}_r \boldsymbol{h}_{t-1}), \tag{2.10}$$

kde $\boldsymbol{W}_r$ je váha současného vstupu $\boldsymbol{x}_t$, $\boldsymbol{U}_r$ je váha výstupu z předešlého modulu $\boldsymbol{h}_{t-1}$ a σ je sigmoidální funkce pro aktualizační bránu.

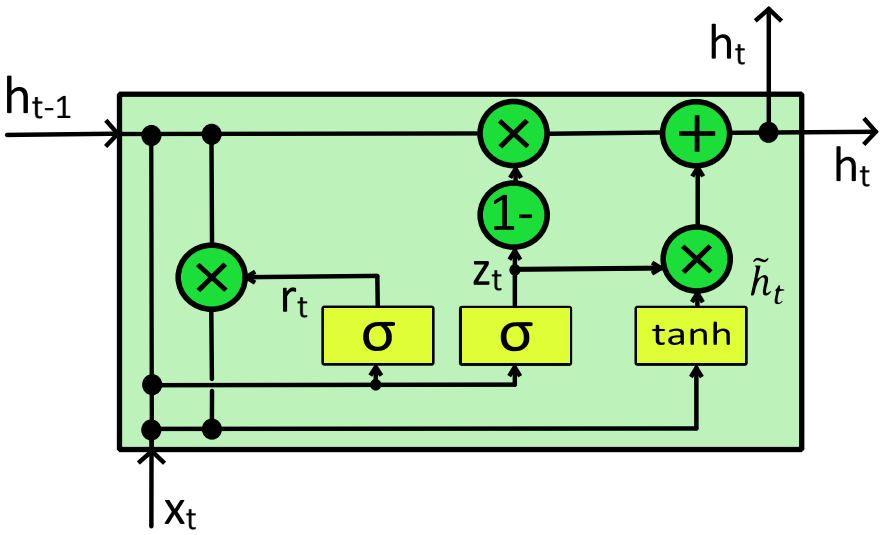
Výstup se nakonec vypočítá pomocí dvou rovnic ovlivněných právě aktualizační bránou a resetující bránou. Výstup prvotně ovlivněný resetující bránou je znázorněn rovnicí:

$$\tilde{\boldsymbol{h}}_t = tanh(\boldsymbol{W}_h \boldsymbol{x}_t + \boldsymbol{r}_t * \boldsymbol{U}_h \boldsymbol{h}_t), \tag{2.11}$$

kde $\boldsymbol{W}_h$ je váha současného vstupu $\boldsymbol{x}_t$, $\boldsymbol{U}_h$ je váha výstupu z předešlého modulu $\boldsymbol{h}_{t-1}$ a operace $*$ slouží k vektorovému násobení.

Konečný výstup je pak dán rovnicí:

$$\boldsymbol{h}_t = \boldsymbol{z}_t * \boldsymbol{h}_{t-1} + (1 - \boldsymbol{z}_t) * \tilde{\boldsymbol{h}}_t, \tag{2.12}$$



Obr. 2.8: Detailní zobrazení vnitřního stavu buňky sítě GRU

3 Detekce sepsí

V roce 2016 byla potřetí přepracována definice pojmu sepse tzv. Sepsis-3. Jedná se o život ohrožující orgánovou dysfunkci, způsobenou deregulovanou odpovědí hostitelského organismu na přítomnost infekce. Sepse se v naší společnosti vyskytuje od začátku naší existence. V poslední letech se vyznačuje vysokou mortalitou, která je i vyšší v porovnání infarktu myokardu.

V České republice je pojem sepse často špatně označován jako "otrava krve". Sepse může přecházet v septický šok, u kterého dochází až k selhání orgánů. Septický šok je definován jako sepse provázená buněčnými, metabolickými a oběhovými abnormalitami, které jsou natolik závažné, že mohou vést ke smrti. Sepse v nemocničních zařízení na JIP stanovištích je detekována pomocí skóre SOFA (Sequential Organ Failure Organ Assessment), které na základě respirace, koagulace, jaterních a renálních funkcí, dále pak stavu vědomí a hemodynamiky určí bodové ohodnocení 0 až 4 body. Pro zrychlení odhalení sepse se stanovuje na pracovištích urgentního příjmu nebo na standartních nemocničních oddělení tzv. qSOFA (quick SOFA), která disponuje pouze tříbodovým systémem $0-3$. Pracuje s parametry jako je systolický krevní tlak ≤ 100 mm Hg, dýchací frekvence ≥ 22 dechů/minutu a změna vědomí. Jestliže jsou pacientovi přiřazeny alespoň 2 body, pak je daný pacient označen za rizikového. [17]

3.1 Metoda podpůrných vektorů

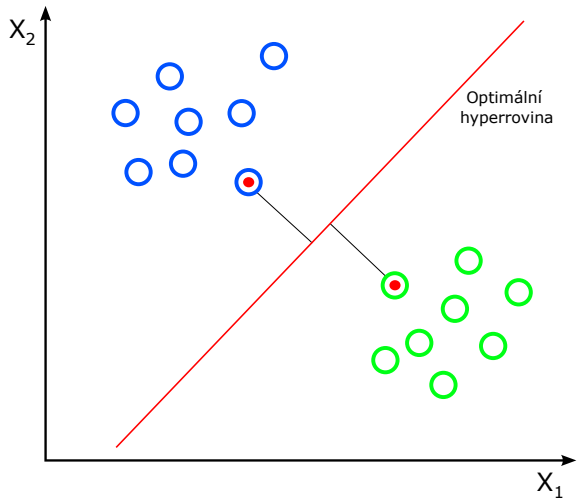
Jedna z metod, která se využívá pro detekci sepse, je metoda podpůrných vektorů (Support Vector Machine - SVM). Jedná se o algoritmus strojového učení s učitelem, který může být použit jak pro klasifikační úlohy, tak pro regresní analýzu. Větší uplatnění nachází pravě při řešení klasifikačních úloh.

Cílem metody SVM je najít hyperrovinu v n-dimenzionálním prostoru (kde n, je počet příznaků daných prvků), která zřetelně klasifikuje jednotlivé prvky. Pro nelineárně oddělitelná data používá techniku nazvanou kernel trik, která najde optimální hranici mezi možnými prvky. Hlavní myšlenkou kernelového triku je používání jádrové funkce, tedy převedení dat do vyšší dimenze, kde už můžeme data rozdělit. Poté provedeme klasifikaci vyhledáním hyperroviny, která velmi dobře odlišuje tyto dvě třídy. Vyhledání optimální hyperroviny můžeme vidět na Obr. 3.1

Metoda podpůrných vektorů byla použita při predikci sepse u pooperačních pacientů. [18] Další systém založený na SVM pro predikci sepse a SIRS z neinvazivní analýzy kardiovaskulárního spektra představil Tang a jeho kolegové. [19]

Jejich experiment začíná popisem databáze MIMIC-II, jde o primární zdroj sběru dat. Dále jsou popsána kritéria klinického zařazení a vyloučení výběru pacientů. Pro

základní odhady byly použity modely SVM a Skryté Markovovy modely (Hidden Markov model - HMM) na datech kontinuální časové řady pro dané pacienty. Bylo zapotřebí použití některých fyziologických hodnot jako např. střední arteriální tlak (SAT), srdeční frekvence (SF) a dechová frekvence (DF).



Obr. 3.1: Na grafu je zobrazena metoda podpůrných vektorů, tedy nejlepší možná hyperrovina, která správně odlišuje dvě různé třídy.

3.2 Markovovy modely

Skryté Markovovy modely (HMM) je jedna z dalších technik pro modelování progrese či predikci onemocnění. [20]. Rané pokusy o modelování progrese sepse založené na Markovových modelech považují pacienty za homogenní skupinu. Avšak většina pacientů se septickým stavem a reakcí na léčbu je často charakterizována významnou heterogenitou jak mezi samotnými pacienty, tak v průběhu času u jednoho pacienta. Současná klinická diagnostická kritéria (rychlé vyhodnocení sekvenčního selhání orgánů (qSOFA)) zohledňují pouze úrovně několika životních funkcí, nikoliv jejich variabilitu, a to obvykle v jediném časovém bodě bez ohledu na minulý stav pacienta. [17]

V tomto projektu autoři zavádí HMM v diskrétním čase, aby analyzovali progresi sepse pacienta. Každý časově diskrétní krok je spjatý s pěti vitálními znaky: systolický krevní tlak, diastolický krevní tlak, srdeční frekvence, dechová frekvence a teplota. Bere se v úvahu heterogenita pacientů, aby určili pravděpodobnost přechodu stavu pacienta.

Použili kohortu 25 000 pacientů s podezřením nebo potvrzenou sepsí. Datový soubor pocházel z Kaiser Permanente v severní Kalifornii. [23] Tato studie ukazuje

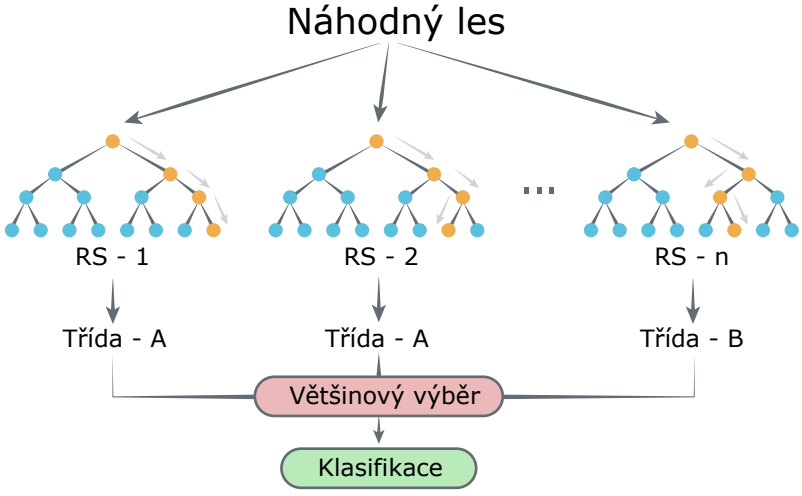
užitečnost HMM při používání v klinické praxi. Slouží k nahlédnutí do základů pacientovi fyziologie, kterou lze použít k informování klinických rozhodnutí a případných zákroků. Tyto modely v kombinaci s použitím dalších fyziologických, laboratorních a léčebných údajů pravděpodobně zlepšily identifikaci vysoce rizikových pacientů, u nichž je indikována smysluplná léčba. Model také indikuje pacienty, kteří mohou být propuštěny z nemocnice, protože se sepse u nich neprojeví.

3.3 Náhodný les

Náhodný les je algoritmus strojového učení, který patří mezi nejpoužívanější algoritmy kvůli jeho jednoduchosti a rozmanitosti. Náhodný les se používá pro klasifikační a regresní úlohy. Skvělé výsledky ukazuje bez jakékoliv optimalizace hyperparametrů, v čem spočívá jeho výhoda. Náhodný les pracuje s jednotlivými rozhodovacími stromy, které spojuje dohromady, aby získal přesnější a stabilnější predikci. Schéma popisující fungování metody náhodného lesa je zobrazeno na Obr. 3.2.

Náhodný les se často používá právě pro predikci sepse. V této práci [21] autor pracuje s náhodným lesem, jakožto modelem vhodným pro predikci. Data, se kterými pracuje, pochází z MIMIC-III (databáze dat). Celkem bylo navrženo 9 modelů, které pracovaly zejména s příznaky jako např. teplota, SAT, SF a DF. Autor řešil problém týkající se chybějících hodnot některých příznaků pomocí zmenšení počtu trénovacích a testovacích dat.

Další modely využívající náhodný les pro predikci sepse využívají v článku [22], kde porovnávají účinnost i jiných vhodných algoritmů strojového učení hodící se na problém včasné predikce či detekce sepse.



Obr. 3.2: Jednoduché schéma fungování algoritmu náhodného lesa, ve kterém jsou součástí rozhodovací stromy (RS).

4 Návrh a implementace

Samotný návrh a jeho implementace probíhala v programovacím prostředí MATLAB R2020b s dodatečnou knihovnou Deep Learning Toolbox™. K uskutečnění praktické části byl využit vzdálený server CUDA3, který disponuje grafickou kartou Nvidia GeForce GTX 1060 (6 GB). Tento server byl poskytnut Ústavem biomedicínského inženýrství - VUT v Brně.

4.1 Popis datasetu

Data, která se v této práci využívají, byla poskytnuta z Challange PhysioNet 2019. [24] Data byla získána ze dvou různých nemocničních zařízení na jednotkách intenzivní péče v USA (Beth Israel Deaconess Medical Center, Emory University Hospital). Každá z těchto nemocnic pracuje rozdílnými systémy pro shromažďování elektronických lékařských záznamů. Sběr dat probíhal v intervalu deseti let se souhlasem příslušných orgánů. Celkově se na sběru dat podílelo přesně 40 336 pacientů. Dataset byl sestaven ze záznamů vitálních funkci, laboratorních hodnot a statistických popisů pacientů. Celkově dataset obsahuje čtyřicet jedna klinických proměnných, z toho je osm životně důležitých, dvacet šest laboratorních a šest demografických. Poslední proměnná je anotace, tedy jestli pacient byl označen za septického či nikoliv. Všechna data z elektronických zdravotnických záznamů prošla mnoha kroky předzpracování. Příznaky všech pacientů byly zhuštěny do hodinových podob (oken). [24]

V datech jsou přiřazeny specifické časové body pro každého septického pacienta podle Sepsis-3:

• $t_{podezřelý}$: Klinické podezření na infekci se identifikovalo podle antibiotik a změření krevních kultur. Jestliže byla podána antibiotika jako první, pak dané krevní kultury musí být získány nejpozději do 24 hodin. Jestliže byly získány krevní kultury, pak musí být podána antibiotika. Pro oba tyto případy platilo, že antibiotika byla podávána nejméně po dobu 72 po sobě následujících hodin.

• t_{SOFA}: Jedná se o poškození orgánů identifikovaný systémem SOFA během 24 hodin.

• t_{sepse}: Začátek sepse byl identifikován pokud:

$$t_{podezřelý} - 24 \leq t_{SOFA} \leq t_{podezřelý} + 12 \tag{4.1}$$

$$t_{sepse} = min(t_{podezřelý}, t_{sepse}) \tag{4.2}$$

Tab. 4.1: Vitální znaky 1-8, 9-34 laboratorní hodnoty

HR	Srdeční frekvence
O2Sat	Pulzní oxymetrie (%)
Temp	Teplota (C)
SBP	Systolický tlak (mm Hg)
MAPA	Střední arteriální tlak (mm Hg)
DBP	Diastolický tlak (mm Hg)
Resp	Rychlost dýchaní
EtCO2	Konečný přílivový oxid uhličitý (mm Hg)
BaseEcess	Míra přebytku hydrogenuhličitanu (mmol/l)
HCO3	Hydrogenuhličitan (mmol / l)
FiO2	Frakce inspirovaného kyslíku (%)
pH	-
PaCO2	Parciální tlak oxidu uhličitého z arteriální krve (mm Hg)
SaO2	Saturace kyslíkem z arteriální krve (%)
AST	Aspartát transamináza (IU / L)
DRDOL	Dusík močoviny v krvi (mg / dl)
Alkalinefos	Alkalická fosfatáza (IU / L)
Vápník	(mg / dl)
Chlorid	(mmol / l)
Kreatinin	(mg / dl)
Bilirubin_direct	Bilirubin přímý (mg / dl)
Glukóza	Glukóza v séru (mg / dl)
Laktát	Kyselina mléčná (mg / dl)
Hořčík	(mmol / dl)
Fosfát	(mg / dl)
Draslík	(mmol / l)
Bilirubin_total	Celkový bilirubin (mg / dl)
TroponinI	Troponin I (ng / mL)
Hct	Hematokrit (%)
Hgb	Hemoglobin (g / dl)
PTT	Částečný tromboplastinový čas (sekundy)
WBC	Počet leukocytů (počet * 10^3 / µL)
Fibrinogen	(mg / dl)
Trombocyty	(počet * 10^3 / µL)

Tab. 4.2: Demografické údaje 35 - 40

Stáří	Roky (100 pro pacienty nad 90 let)
Rod	Žena (0) nebo Muž (1)
Oddíl 1	Administrativní identifikátor pro jednotku JIP
Oddíl 2	Administrativní identifikátor pro jednotku JIP
HospAdmTime	Hodiny mezi přijetím do nemocnice a hospitalizací na JIP
ICULOS	Délka pobytu na JIP (hodiny od přijetí na JIP)

Tab. 4.3: Přiřazení anotace pro septické a neseptické pacienty. Jedná se tak o poslední proměnou v datech.

Anotace	K septickým pacientům je přiřazena 1, jestliže $t \geq t_{sepse} - 6$, a 0 jestliže $t < t_{sepse} - 6$. Pro neseptické pacienty je přiřazena 0.

4.2 Předzpracování dat

Práce s daty je nedílnou součástí uskutečnění kvalitního modelu. Předzpracování zahrnovalo rozdělení dat, řešení problému s chybějícími hodnotami, škálování dat a vyřešení nerovnoměrného zastoupení jednotlivých predikčních skupin.

Data bylo nutné rozdělit na dvě separované skupiny kvůli následnému tréninku a testování modelu. Toto rozdělení bylo provedeno náhodně a v poměru 9:1 na data trénovací a testovací. Nutno říci, že žádný pacient nebyl společně využíván, jak v množině určené k trénování, tak k testování a zároveň všichni pacienti byli přiřazeni k jednotlivé množině právě jednou.

V datech se vyskytovalo velké množství chybějících hodnot NaN. Existuje více způsobů, jak nahradit chybějící hodnoty. Jeden ze způsobů může být nahrazení poslední známou hodnotou. Další variantou je nahrazení průměrnou hodnotou či nulou. V mé práci pracuji zejména s přepisem chybějících hodnot NaN na 0. Orientační množství chybějících hodnot vyjadřuje Tab. 4.4.

Velmi podstatnou částí je škálování dat. Škálování je důležité kvůli převedení všech hodnot příznaků do stejného měřítka. Poté každá hodnota daného příznaku přispívá stejnou měrou při řešení určitého problému. V mém případě jsem prováděl standardizaci pomocí směrodatné odchylky. Tato standardizace je často označována jako z-skóre, které je popsáno rovnicí:

$$z = \frac{x - \mu}{\sigma}, \tag{4.3}$$

kde x je jednotlivá hodnota příznaku z dat, μ je průměr hodnot daného příznaku z dat, σ je směrodatná odchylka. Standardizace u trénovací množiny byla prove-

Tab. 4.4: Tabulka popisuje srovnání chybějících hodnot NaN vztažené na počet příznaků všech pacientů. Celkem bylo použito 40 příznaků pro každého pacienta. Tedy v průměru pro všechny pacienty u 28 příznaků chybělo více než 50% hodnot.

Chybějící hodnoty [%]	>25%	>50%	>75%
Počet příznaků	31	28	27

Tab. 4.5: Srovnání nerovnoměrného zastoupení septických a neseptických pacientů v rámci různých nemocničních zařízení, ze kterých data pocházela.

Nemocniční systém	A	B	CELKEM
Počet pacientů	20 336	20 000	40 336
Počet neseptických pacientů	18 546	18 858	37 404
Počet septických pacientů	1 790	1 142	2 932

dena přes všechny pacienty. Stejným způsobem byla uskutečněna standardizace pro testovací data.

Poslední problém, který jsem při předzpracování dat řešil, byl nevyvážený dataset. Jestliže bych tento problém nijak neřešil, došlo by k vytvoření špatného modelu, který by špatně klasifikoval/predikoval jednotlivé třídy. Jednou z metod, který tento problém řeší, je rozkopírování méně zastoupené třídy. Další řešení může spočívat ve vymazání naopak více zastoupené třídy. Rozhodl jsem se ponechat dataset tak, jak jsem ho obdržel, ovšem jednotlivým třídám jsem přiřadil určitou váhu podle poměru zastoupení. Nerovnoměrné zastoupení jednotlivých tříd vyjadřuje Tab. 4.5.

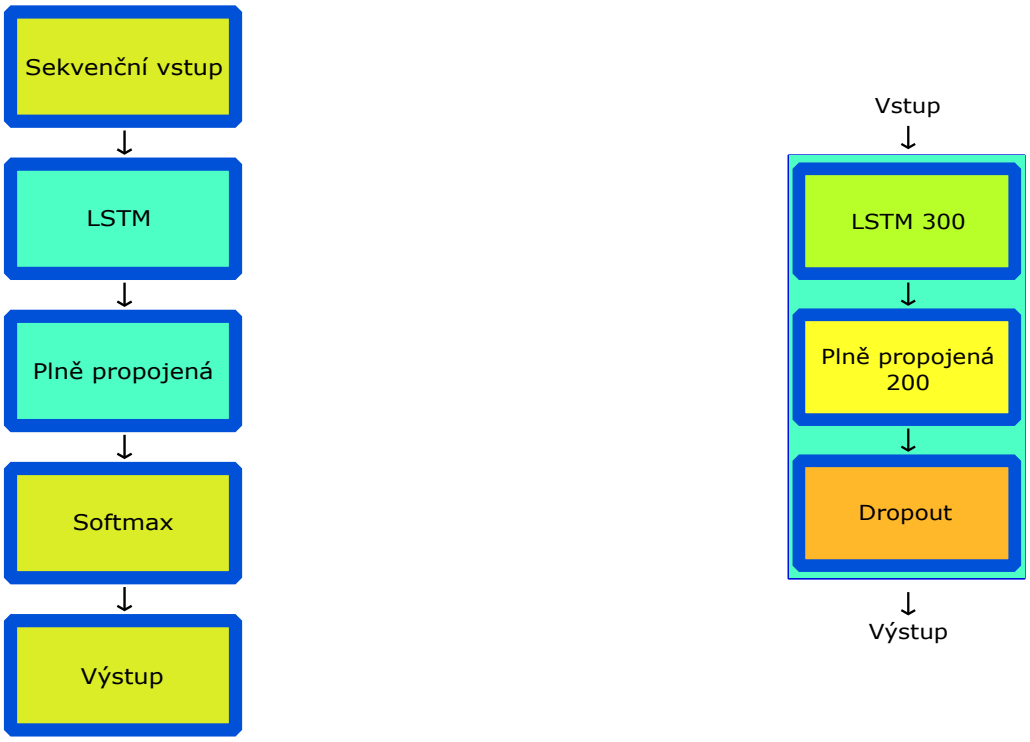
4.3 Implementace sítě LSTM

Jako první metodu pro predikování sepsí z klinických signálů jsem si vybral rekurentní síť LSTM. Síťe LSTM, které jsou více popsány v kapitole 2.3.1, jsou vynikajícím nástrojem právě pro predikci. Predikování různých onemocnění nachází v posledních letech velké uplatnění v medicíně.

Rozhodl jsem se vytvořit a následně naučit celkem pět LSTM sítí. Jednotlivé sítě se lišily v počtu vrstev, jenž obsahovaly. Obecnou architekturu, která byla použita, popisuje Obr. 4.1a. Všechny vytvořené sítě pracovaly se sekvenčními vstupy a jejich výstupem byla opět sekvence. Mluvíme tedy o režimu sítě - sekvence k sekvenci.

Zpočátku bylo nutné nastavení vstupní vrstvy, jelikož jsem použil všechny příznaky daných pacientů. Velikost vstupní vrstvy byla nastavena na 40 neuronů. Počet vnitřních buněk u jednotlivých LSTM vrstev byl nastaven na hodnotu 300. Jednotlivé plně propojené vrstvy byly nastaveny na hodnotu 200 neuronů. Další vrstva,

kterou jsem se rozhodl implementovat do modelu, byla dropout nastavena na hodnotu 0,5.



(a) Obecná architektura

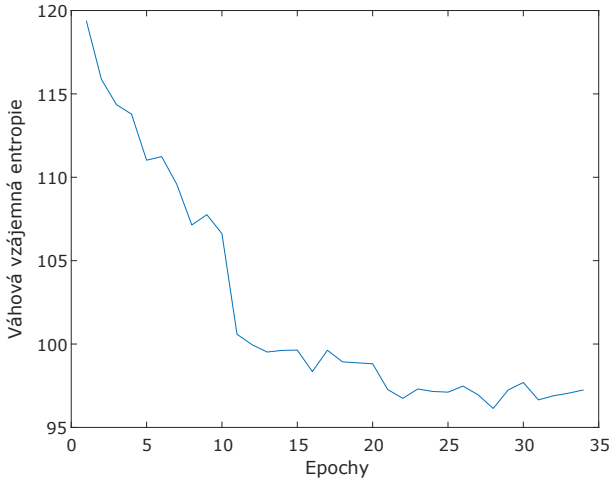
(b) Konkrétní vrstvy sítě

Obr. 4.1: Schéma 4.1a nám ukazuje obecnou architekturu LSTM sítě, která byla použita jako inspirace pro výslednou síť. Schéma 4.1b ukazuje konkretní použití jednotlivých vrstev, které byly ve finální implementaci poskládány za sebe podle počtu LSTM vrstev.

K samotnému tréninku sítě byl použit optimalizační algoritmus ADAM, kde hodnoty jeho hyperparametrů byly nastaveny na $\beta_1 = 0,9$ a $\beta_2 = 0,999$. Dále bylo potřeba nastavit konstantu učení, která byla nastavena na hodnotu 10^{-4} s tím, že každých 10 epoch docházelo ke snížení exponenciálu o hodnotu 1. Maximální počet epoch byl nastaven na hodnotu 35. Použitá byla i L2 regularizace s konstatnou 10^{-8} a oříznutí přechodu gradientu. Rozdílná délka vstupních signálů byla řešena pomocí doplnění nul (zero-padding). Dávka učení (batch) byla nastavena na hodnotu 70. Jako ztrátová funkce byla použita váhová vzájemná entropie, kvůli nerovnoměrnému zastoupení jednotlivých tříd (septici/neseptici). Váhová vzájemná entropie je popsána rovnicí:

$$E_{wce} = -\frac{1}{M}\sum_{m=1}^{M}[w \times y_m \times \log(h_\theta(x_m)) + (1 - y_m) \times \log(1 - h_\theta(x_m))], \qquad (4.4)$$

kde w je váha, y_m je skutečná hodnota výstupu z tréninku sítě, x_m jsou vstupy příznaků určené pro trénink sítě, h_θ je model s váhami neuronové sítě. Ukázka učení LSTM je zobrazena na Obr. 4.2.



Obr. 4.2: Ukázka učení LSTM sítě, tedy průběh její chybové funkce.

4.4 Implementace konvoluční sítě

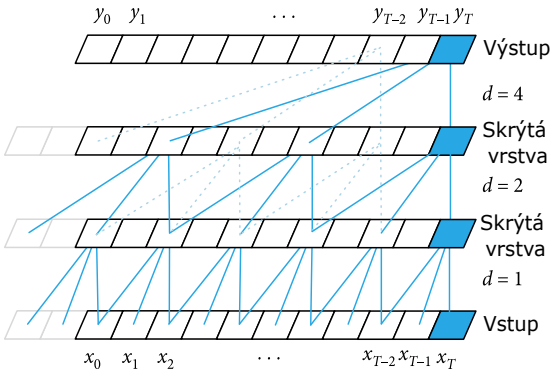
Druhá metoda, která byla pro zhotovení práce využita, byla časová konvoluční síť (TCN). [30] Jedná se o síť, která se v poslední době hojně využívá na sekvenční data. Základem časové konvoluční sítě je vrstva rozšířené kauzální konvoluce. Aktivace se tedy vypočítává pro konkretní časový krok a v budoucích krocích se již neprojevuje. Aby došlo ke spojení předešlých časových kroků, tak se tyto vrstvy skládají na sebe. Důležitou roli zde hraje dilatační faktor, který se exponenciálně zvyšuje s tím, kolik je nad sebou poskládáno vrstev. Dilatační faktor je součástí sítě, a to z důvodu důležitosti pokrytí velkého pracovního prostoru sítě. Jestliže dilatační faktor $k-t$é konvoluční vrstvy je 2^{k-1} a krok je 1, pracovní pole sítě se vypočítá pomocí rovnice:

$$R = (f-1)(2^K - 1) + 1, \tag{4.5}$$

kde f je velikost filtru, K je počet konvolučních vrstev.

Na TCN tedy můžeme pohlížet jako na kombinaci 1D plně propojené sítě a kauzální konvoluce s přidáním prvku dilatace. Síť opět byla nastavena pro režim - sekvence k sekvenci, tedy vstup i výstup byla sekvence.

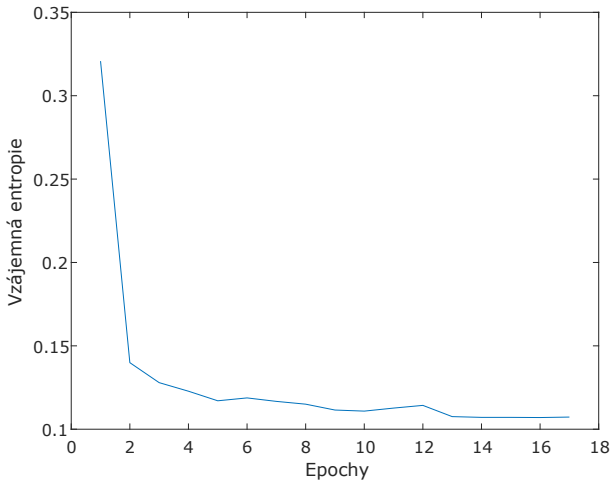
Při návrhu jsem se rozhodl použít celkem 4 bloky obsahující vrstvy rozšířené kauzální konvoluce s 160 filtry o velikosti 40. Každý z těchto bloků tedy obsahoval 2 vrstvy rozšířené kauzální konvoluce a normalizace vah. Ke každé byla přidána



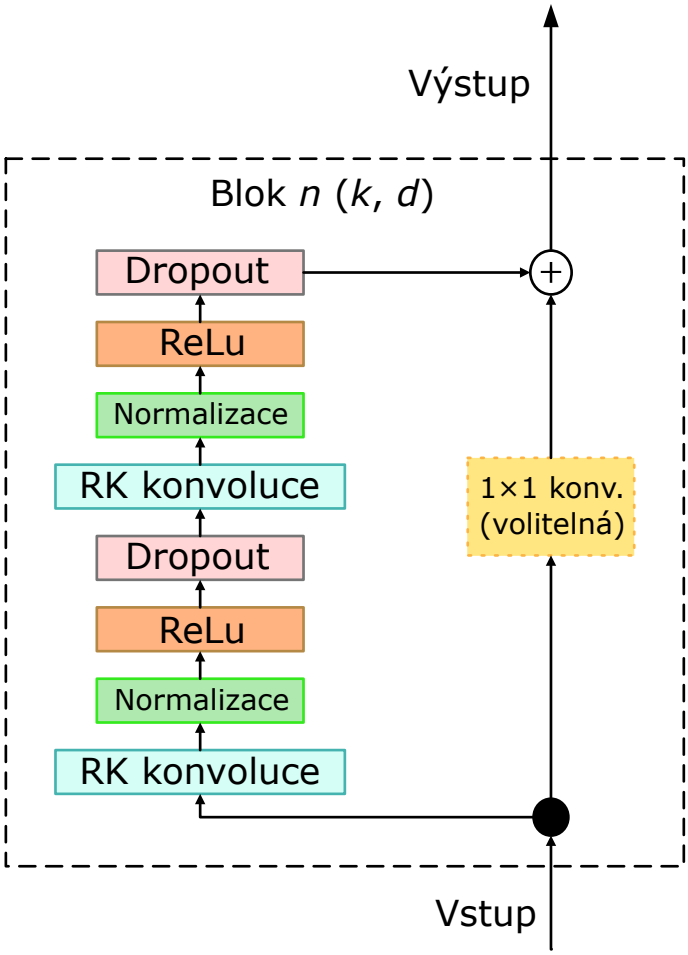
Obr. 4.3: Obecné zobrazení architektury rozšířené kauzální dilatace, pro dilatační faktor $d = 1, 2, 3$. Upraveno podle [30].

vrstva ReLu kvůli zachování nelinearity a vrstva dropout nastavena na hodnotu 0,05. Součástí je i spojení s 1×1 konvolucí, která se používá při rozdílných rozměrech vstupních a výstupních dat. Samotné schéma nám prezentuje Obr. 4.5.

Jako optimalizační algoritmus byl zvolen ADAM. Konstanta učení byla nastavena na hodnotu 10^{-3} s tím, že každých 10 epoch docházelo ke snížení exponenciálu o hodnotu 1. Maximální počet epoch byl nastaven na hodnotu 35. Součástí byla i L2 regularizace a oříznutí velkých přechodů gradientu. Dávka učení (batch) byla nastavena na hodnotu 70. Jako ztrátová funkce byla využita vzájemná entropie. Ukázka učení TCN je zobrazena na Obr. 4.4



Obr. 4.4: Ukázka učení časové konvoluční neuronové sítě, tedy průběh její chybové funkce.



Obr. 4.5: Schéma pro jeden blok použité TCN, kde RK konvoluce představuje rozšířenou kauzální konvoluci. Součástí schématu je i 1×1 konvoluce, která se používá při rozdílných rozměrech vstupních a výstupních dat.

5 Vyhodnocení výsledků

5.1 Statistické metriky

Pro stejnorodé vyhodnocení všech modelů jsem se zaměřil zejména na výpočet senzitivity, pozitivní prediktivní hodnoty a dice koeficientu. Jelikož se jednalo o data, která byla poskytnuta z Challange PhysioNet, tak jako další srovnávací metrika, bylo použito utility score.

Senzitivita (Recall) je pravděpodobnost, že výsledek bude pozitivní, pokud je osoba skutečně septická. Je popsána rovnicí:

$$Se = \frac{TP}{TP + FN}, \tag{5.1}$$

kde TP je skutečně pozitivní a FN je falešně negativní.

Pozitivní prediktivní hodnota (Precision) je pravděpodobnost, že pacient je skutečně septický, když výsledek metody je pozitivní. Je popsána rovnicí:

$$PPV = \frac{TP}{TP + FP}, \tag{5.2}$$

kde TP je skutečně pozitivní a FP je falešně pozitivní.

Dice koeficient (F1-score) je jakýsi kompromis mezi senzitivitou a pozitivní prediktivní hodnotou, jde tedy o harmonický průměr těchto koeficientů. Dochází ke srovnání skutečných hodnot a predikčních hodnot - míra shody. Je popsán rovnicí:

$$DSC = \frac{2TP}{2TP + FP + FN}, \tag{5.3}$$

kde TP je skutečně pozitivní, FP je falešně pozitivní a FN je falešně negativní.

Utility score je speciální statistická metrika vytvořená přímo pro Cardiology Challenge PhysioNet 2019. Jedná se o metriku, která hodnotí predikci sepse v různých časových pásmech. Utility score je tedy penalizační metrika pro pozdní nebo nedetekovanou predikce sepse u septických pacientů a predikci sepse u neseptických pacientů. Naopak bonusové body jsou přiřazeny pro včasnou predikci sepse u septických pacientů. Algoritmus tedy vytváří jednotlivé skóre pro pacienta přes jeho všechna časová pásma (hodinová okna měření). Následně dochází k agregaci všech skóre pacienta a přes všechny pacienty. Tím je vypočteno konečné skóre pro datovou sadu.

Nechť $x(s,t) = 1$ indikuje pozitivní predikci sepse pro pacienta s v čase t, jinak $x(s,t) = 0$. Nechť $\delta(s) = 1$, jestliže je pacient s eventuálně septický, jinak $\delta(s) = 0$.

Definice utility score:

$$U(s,t) = \begin{cases} U_{TP}(s,t), & x(s,t) = 1, \delta(s) = 1, \\ U_{FP}(s,t), & x(s,t) = 1, \delta(s) = 0, \\ U_{FN}(s,t), & x(s,t) = 0, \delta(s) = 1, \\ U_{TN}(s,t), & x(s,t) = 0, \delta(s) = 0, \end{cases} \tag{5.4}$$

Celkové utility score je dáno rovnicí:

$$U_{celkove} = \sum_{s \in S} \sum_{t \in T(s)} U(s,t) \tag{5.5}$$

Normalizované utility score je dáno rovnicí:

$$U_{normalizovane} = \frac{U_{celkove} - U_{bezpredikce}}{U_{optimalni} - U_{bezpredikce}} \tag{5.6}$$

Jestliže model predikoval sepsi mezi 12 hodinami před a 3 hodinami po t_{sepse} u septických pacientů, tak utility score se projevilo bonusovými body (nejvíce) $1,0$. Penalizace se projevila u predikce sepse dříve než 12 hodin před časem t_{sepse} penalizací (nejvíce) $0,05$. Penalizace nastala i v případě vůbec nedetekované sepse, a to $-2,0$ body.

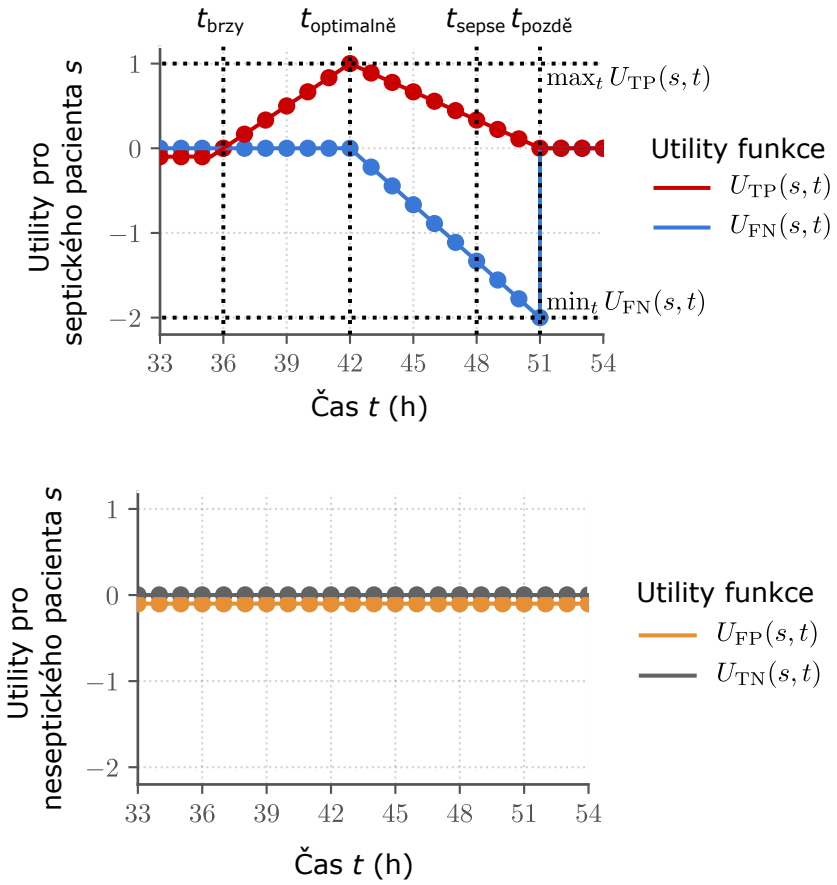
Penalizace body (nejvíce) $0,05$ byla přičtena i v případě falešného poplachu, tedy když model predikoval sepsi u neseptického pacienta.

Grafy popisující utility score jsou zobrazeny na Obr. 5.1.

5.2 Optimální práh

Stanovení prahové hodnoty souvisí s finálním zařazení pacientů do jednotlivých tříd septický/neseptický.

Hledání optimálního prahu probíhalo pomocí genetického algoritmu. [29] Genetický algoritmus (GA) je metoda řešení omezených i neomezených optimalizačních problémů na základě procesu přirozeného výběru. GA napodobují chovaní přirozené evoluce. V mém případě algoritmus úzce pracoval s funkcí utility score a vypočítával největší skóre pro daný práh. Jedná se o iterační proces, tedy v každé iteraci dochází k úpravě prahu. Ukončovací podmínka algoritmu může být maximální počet iterací nebo nalezení nejvyšší hodnoty skóre, pro konkrétní hodnotu prahu.



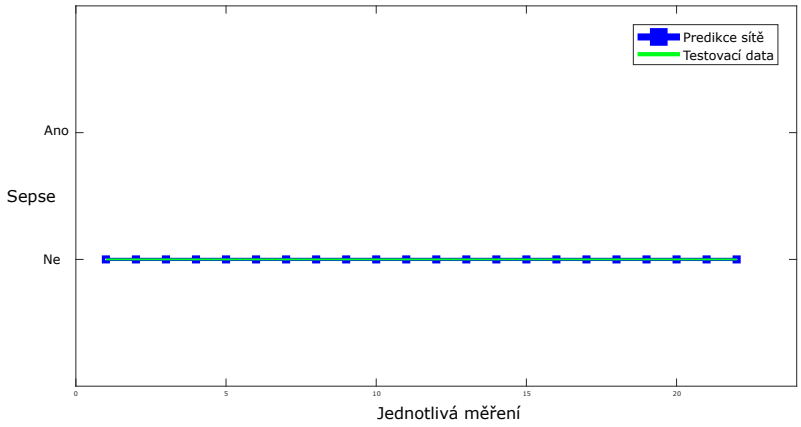
Obr. 5.1: Ukázka grafů pozitivních i negativních predikcí. Horní graf reprezentuje utility score pro septické pacienty, spodní graf reprezentuje utility score pro neseptické pacienty. Jako příklad je uvedena doba $t_{sepse} = 48h$ nástupu sepse. Upraveno podle [24].

Tab. 5.1: Dosažené výsledky pro všechny implementované modely, které byly vyhodnoceny pomocí senzitivity, pozitivní prediktivní hodnoty, dice koeficientu a speciální metriky utility score.

Model	Se	PPV	DSC	Utility score
LSTM 3 vrstvy	0,559	0,067	0,120	0,345
LSTM 4 vrstvy	0,550	0,074	0,130	0,354
LSTM 5 vrstvy	0,504	0,081	0,139	0,346
LSTM 6 vrstvy	0,567	0,066	0,118	0,341
LSTM 7 vrstvy	0,526	0,079	0,137	0,356
TCN	0,638	0,067	0,121	0,377

5.3 Dosažené výsledky

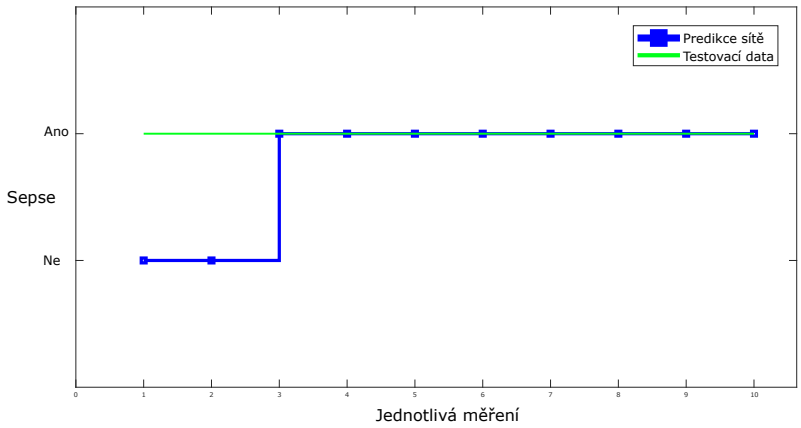
Finální podoba dosažených výsledků je zaznamenána v Tab. 5.1. Celkem bylo implementováno pět sítí LSTM, které se lišily v počtu vrstev a jedna síť TCN. K vyhodnocení byly použity statistické metriky popsány v kapitole 5.1. Příklady některých predikcí sepsí jsou představeny na obrázcích 5.2, 5.3, 5.4, 5.5, 5.6 a 5.7.



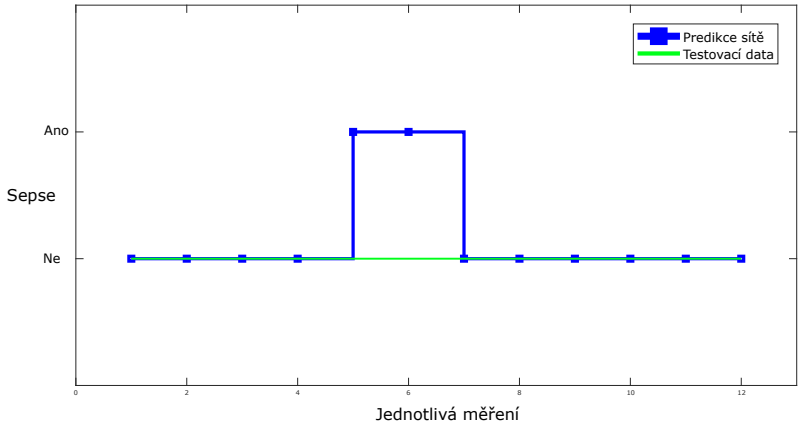
Obr. 5.2: Predikce LSTM sítě vybraného pacienta č. 3698 z testovacích dat.

5.4 Diskuze

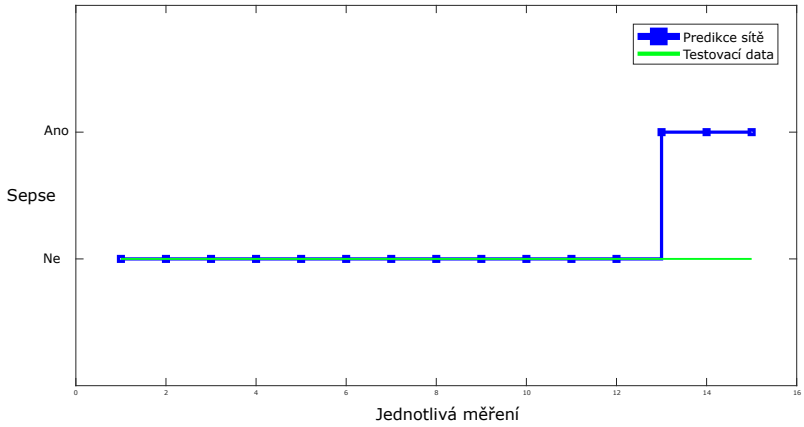
Jak můžeme vidět v Tab. 5.1, nejlepší výsledek získala síť TCN, která dosahuje nejvyšší hodnoty normalizovaného utility score. Jako druhá v pořadí se umístila síť LSTM (7), která se skládala ze sedmi vrstev. Jestliže se zaměříme na koeficient dice,



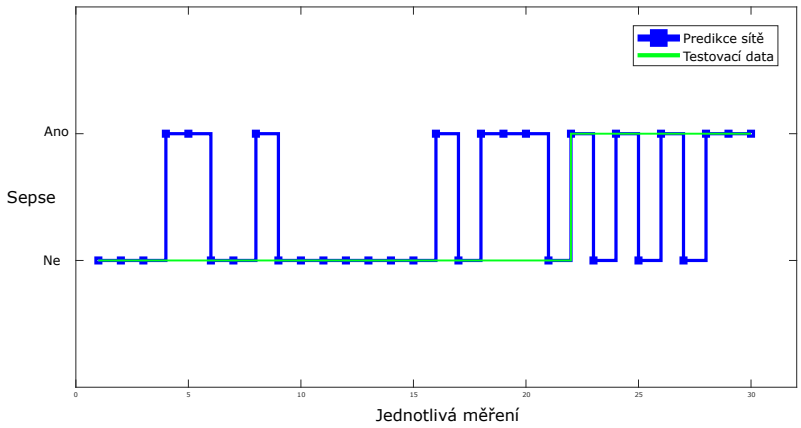
Obr. 5.3: Predikce LSTM sítě u vybraného pacienta č. 2895 z testovacích dat.



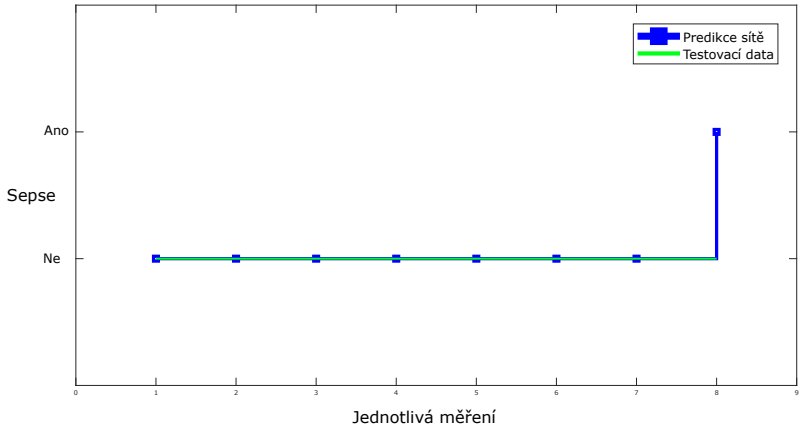
Obr. 5.4: Predikce LSTM sítě u vybraného pacienta č. 996 z testovacích dat.



Obr. 5.5: Predikce TCN sítě vybraného pacienta č. 99 z testovacích dat.



Obr. 5.6: Predikce TCN sítě u vybraného pacienta č. 193 z testovacích dat.



Obr. 5.7: Predikce TCN sítě u vybraného pacienta č. 274 z testovacích dat.

tak nejlepší hodnocení dosáhla síť LSTM se čtyřmi vrstvami, ovšem LSTM (7) se sedmi vrstvami se umístila hned na druhém místě.

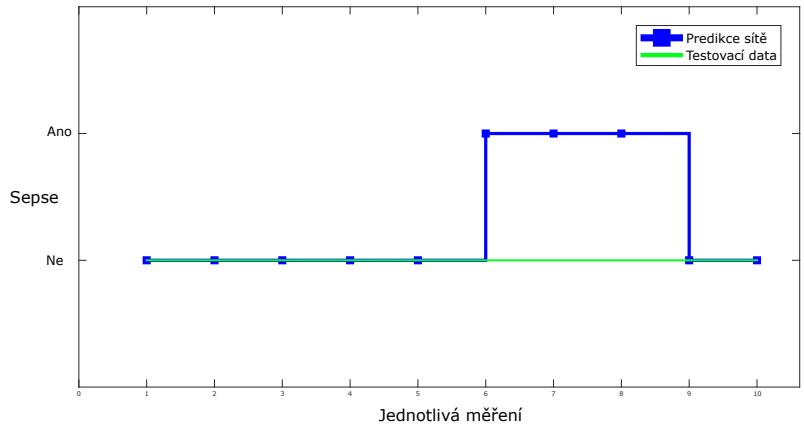
V této části diskuze bych se rád zaobíral srovnáním navržených sítí, konkrétně těm, které dosahovaly nejlepších výsledků, a to sedmivrstvá LSTM (7) a TCN.

První výhodu LSTM (7) shledávám právě při jednodušší implementaci vzhledem k využívání Deep Learning Toolbox™, který je součástí programovacího prostředí MATLAB. Jako další z výhod spatřuji rychlost trénování. Síť LSTM (7) se trénovala v řádu dvou až čtyř hodin, což je oproti TCN polovina až třetina času tréninku. Navzdory tomuto času síť LSTM (7) dosahuje přijatelných výsledků.

Jak už bylo řečeno, nevýhodou u práce se sítí TCN byl jednoznačně čas jejího tréninku, který dosáhl až k třinácti hodinám. Jako velmi pozitivní spatřuji zejména nejlepší dosažené výsledky při predikování sepsí z klinických signálů. Další výhodou bylo zejména nahlédnutí do přesného fungování této sítě a z části odhalení tzv. černé

schránky, kde bylo možné vidět, jak se síť doopravdy učí.

Obě sítě predikovaly velké množství falešně pozitivních výsledků, a to i tehdy, když pacient byl označen za neseptického celou dobu jeho měření. V praxi by tedy tento problém znamenal to, že by pacientovi byly podány určité medikamenty. Ovšem toto všechno by bylo zbytečné, jelikož by pacientovi sepse vůbec nenastala. Mluvíme tedy o falešném poplachu. Tento problém se nejvíce odrazil na hodnotách statických vyhodnocení. Příklad takové falešně pozitivní predikce je možné vidět na Obr. 5.8.



Obr. 5.8: Ukázka falešné pozitivní predikce u sítě LSTM(7) z testovacích dat.

Jak už bylo naznačeno v kapitole 4.1, otázkou umělé inteligence pro predikování sepsí z klinických signálů s těmito poskytnutými daty se zabývalo několik desítek týmů po celém světě. Rozhodl jsem se zde uvést i srovnání dosažených výsledků s některými týmy. Hlavní metrikou srovnání je dříve zmíněné utility score v kapitole 5.1. Týmy pracovaly s různými metodami umělé inteligence. Vítězným týmem celé soutěže se stal tým uvedený jako T4 v Tab. 5.2, který dosáhl na svých validačních datech hodnotu utility score 0,430. [28]. Jestliže porovnám výsledky, které jsem dosáhl při implementaci mých modelů a výsledky, které získali odborníci z celého světa, tak na validačních datech, ze kterých jsme vycházeli, mohu říci, že se těmto výsledkům z části přibližuji.

Na základě zkušeností, o které mě práce obohatila, bych se momentálně snažil klást větší důraz na předzpracování dat či výběr nejdůležitějších příznaků pomocí dalších metod umělé inteligence. Tento postup by mohl pomoci ke zlepšení predikování mých modelů a tím pádem k lepším výsledkům. Dále by bylo zajímavé více experimentovat s architekturami a kombinovat LSTM a konvoluční sítě, což by mohla být s velkou pravděpodobností další cesta ke zlepšení výsledků.

Tab. 5.2: Výsledky některých týmu, které zpracovávaly včasnou predikci sepsí z klinických dat jako Challange PhysioNet 2019. [25] [26] [27] [28]

Týmy	**Utility score**
T1: Luan Tran , Manh Nguyen , Cyrus Shahabi	0,363
T2: John Anda Du, Nadi Sadr, Philip de Chazal	0,403
T3: Morteza Zabihi , Serkan Kiranyaz , Moncef Gabbouj	0,428
T4: James Morrill, Andrey Kormilitzin, Alejo Nevado-Holgado, Sumanth Swaminathan, Sam Howison, Terry Lyons	0,430
Vlastní implementace	**Utility score**
LSTM (7)	0,356
TCN	0,377

Závěr

Tématem této bakalářské práce bylo použití umělé inteligence pro predikování sepsí z klinických signálů. Z prvopočátku bylo nutné se seznámit s problematikou umělé inteligence a jejími metodami. Na základě toho byla vypracována literární rešerše, jejíž první část byla soustředěna na základní principy, fungování a učení neuronových sítí. Druhá část se zabývala hlubokými neuronovými sítěmi, kde největší důraz byl kladen na sítě rekurentní. Třetí část se věnovala dalším možným metodám umělé inteligence pro řešení otázky včasné predikce sepse.

Z výše uvedené literární rešerše byly vybrány dvě vhodné metody, které byly implementovány v programovém prostředí MATLAB verze R2020b. První vhodnou metodou, která by řešila problém včasné predikce, byla zvolena metoda LSTM. Síť LSTM patří do skupiny rekurentních neuronových sítí, které se hodí pravě na práci z časovými sekvencemi. Další metodou byla zvolena časová konvoluční síť TCN.

V následném postupu praktické části dochází k vyhodnocení jednotlivých výsledků pomocí vhodných statistických metrik, kterými jsou: senzitivita, pozitivní prediktivní hodnota a dice koeficient. Nedílnou součástí je srovnávací metrika utility score, která sloužila k srovnání výsledků s týmy z celého světa. Nejlepší hodnoty normalizovaného utility score dosáhla časová konvoluční síť TCN, a to 0,377. Na druhém místě se umístila síť LSTM se sedmi vrstvami s hodnotou normalizovaného utility score 0,356. Nejvyšší hodnoty dice koeficientu dosáhla LSTM s pěti vrstvami. Nejlepší dosažené výsledky LSTM (7) a TCN, včetně jednotlivých predikcí modelů, jsou v závěru práce porovnávány s výsledky některých týmů, které řešily stejnou otázku, jenž je tématem této bakalářské práce. Samotný závěr je věnován úvaze, která by mohla sloužit ke zlepšení dosavadních výsledků.

Literatura

[1] MARINI, F., R. BUCCI, A.L. MAGRÌ a A.D. MAGRÌ. Artificial neural networks in chemometrics: History, examples and perspectives. *Microchemical Journal* [online]. 2008, **88**(2), 178-185 [cit. 2019-12-04]. DOI: 10.1016/j.microc.2007.11.008. ISSN 0026265X.

[2] GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. *Deep learning*. Cambridge, Massachusetts: The MIT Press, [2016]. ISBN isbn:0262035618.

[3] JAN, Jiří. *Číslicová filtrace, analýza a restaurace signálů*. 2. upr. a rozš. vyd. Brno: VUTIUM, 2002. ISBN 80-214-1558-4.

[4] YEUNG, Daniel S., Ian CLOETE, Daming SHI a Wing W. Y. NG. *Sensitivity Analysis for Neural Networks* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010 [cit. 2019-12-04]. Natural Computing Series. DOI: 10.1007/978-3-642-02532-7. ISBN 978-3-642-02531-0.

[5] YUAN, Zihao, Ji LI, Zhe LI, et al. Softmax Regression Design for Stochastic Computing Based Deep Convolutional Neural Networks. In: *Proceedings of the on Great Lakes Symposium on VLSI 2017 - GLSVLSI '17* [online]. New York, New York, USA: ACM Press, 2017, 2017, s. 467-470 [cit. 2019-12-04]. DOI: 10.1145/3060403.3060467. ISBN 9781450349727.

[6] PHAM, Vu, Theodore BLUCHE, Christopher KERMORVANT a Jerome LOURADOUR. Dropout Improves Recurrent Neural Networks for Handwriting Recognition. In: *2014 14th International Conference on Frontiers in Handwriting Recognition* [online]. IEEE, 2014, 2014, s. 285-290 [cit. 2019-12-04]. DOI: 10.1109/ICFHR.2014.55. ISBN 978-1-4799-4334-0.

[7] BAYAR, Belhassen a Matthew C. STAMM. A Deep Learning Approach to Universal Image Manipulation Detection Using a New Convolutional Layer. In: *Proceedings of the 4th ACM Workshop on Information Hiding and Multimedia Security - IHMMSec '16* [online]. New York, New York, USA: ACM Press, 2016, 2016, s. 5-10 [cit. 2019-12-09]. DOI: 10.1145/2909827.2930786. ISBN 9781450342902.

[8] SCHERER, Dominik, Andreas MÜLLER a Sven BEHNKE. Evaluation of Pooling Operations in Convolutional Architectures for Object Recognition. DIAMANTARAS, Konstantinos, Wlodek DUCH a Lazaros S. ILIADIS, ed. *Artificial Neural Networks — ICANN 2010* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, 2010, s. 92-101 [cit. 2019-12-04]. Lecture Notes in Computer Science. DOI: 10.1007/978-3-642-15825-4_10. ISBN 978-3-642-15824-7.

[9] LI, Jing, Ji-hang CHENG, Jing-yuan SHI a Fei HUANG. Brief Introduction of Back Propagation (BP) Neural Network Algorithm and Its Improvement. JIN, David a Sally LIN, ed. *Advances in Computer Science and Information Engineering* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, 2012, s. 553-558 [cit. 2019-12-04]. Advances in Intelligent and Soft Computing. DOI: 10.1007/978-3-642-30223-7_87. ISBN 978-3-642-30222-0.

[10] BOTTOU, Léon. Stochastic Gradient Descent Tricks. MONTAVON, Grégoire, Geneviève B. ORR a Klaus-Robert MÜLLER, ed. *Neural Networks: Tricks of the Trade* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, 2012, s. 421-436 [cit. 2019-12-04]. Lecture Notes in Computer Science. DOI: 10.1007/978-3-642-35289-8_25. ISBN 978-3-642-35288-1.

[11] RUDER, Sebastian. *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION* [online]. 15 Jun 2017 [cit. 2019-12-09]. Dostupné z: https://arxiv.org/pdf/1609.04747.pdf

[12] QIAN, Ning. On the momentum term in gradient descent learning algorithms. *Neural Networks* [online]. 1999, **12**(1), 145-151 [cit. 2019-12-09]. DOI: 10.1016/S0893-6080(98)00116-6. ISSN 08936080.

[13] Diederik P. Kingma a Jimmy Lei Ba. *ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION* [online]. 30. Jan 2017 [cit. 2019-12-09]. Dostupné z: https://arxiv.org/pdf/1412.6980.pdf

[14] *Investigacion agraria: Sistemas y recursos forestales*. Vol.1,(1992) č.1. Madrid: Instituto Nacional de Investigación y Tecnologia Agraria y Alimen., 1992.

[15] ZHOU, Guo-Bing, Jianxin WU, Chen-Lin ZHANG a Zhi-Hua ZHOU. Minimal gated unit for recurrent neural networks. *International Journal of Automation and Computing* [online]. 2016, **13**(3), 226-234 [cit. 2019-12-04]. DOI: 10.1007/s11633-016-1006-2. ISSN 1476-8186.

[16] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho a Yoshua Bengio. *Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling* [online]. 11. Dec 2014 [cit. 2019-12-09]. Dostupné z: https://arxiv.org/pdf/1412.3555.pdf

[17] GYAWALI, Bishal, Karan RAMAKRISHNA a Amit S DHAMOON. Sepsis: The evolution in definition, pathophysiology, and management. *SAGE Open Medicine* [online]. 2019, **7** [cit. 2019-12-04]. DOI: 10.1177/2050312119835043. ISSN 2050-3121.

[18] Kim J, Blum J, Scott C. *Temporal features and kernel methods for predicting sepsis in postoperative patients. Technical Report.* Ann Arbor: University of Michigan; 2010.

[19] TANG, Collin H H, Paul M MIDDLETON, Andrey V SAVKIN, Gregory S H CHAN, Sarah BISHOP a Nigel H LOVELL. Non-invasive classification of severe sepsis and systemic inflammatory response syndrome using a nonlinear support vector machine: a preliminary study. *Physiological Measurement* [online]. 2010, **31**(6), 775-793 [cit. 2021-4-26]. ISSN 0967-3334. Dostupné z: doi:10.1088/0967-3334/31/6/004

[20] JACKSON, Christopher H., Linda D. SHARPLES, Simon G. THOMPSON, Stephen W. DUFFY a Elisabeth COUTO. Multistate Markov models for disease progression with classification error. *Journal of the Royal Statistical Society: Series D (The Statistician)* [online]. 2003, **52**(2), 193-209 [cit. 2021-4-26]. ISSN 00390526. Dostupné z: doi:10.1111/1467-9884.00351

[21] DARWICHE, Aiman a Sumitra MUKHERJEE. Machine Learning Methods for Septic Shock Prediction. In: *Proceedings of the 2018 International Conference on Artificial Intelligence and Virtual Reality - AIVR 2018* [online]. New York, New York, USA: ACM Press, 2018, 2018, s. 104-110 [cit. 2021-4-27]. ISBN 9781450366410. Dostupné z: doi:10.1145/3293663.3293673

[22] TAYLOR, R. Andrew, Joseph R. PARE, Arjun K. VENKATESH, Hani MOWAFI, Edward R. MELNICK, William FLEISCHMAN, M. Kennedy HALL a Alan JONES. Prediction of In-hospital Mortality in Emergency Department Patients With Sepsis: A Local Big Data-Driven, Machine Learning Approach. *Academic Emergency Medicine* [online]. 2016, **23**(3), 269-278 [cit. 2021-4-27]. ISSN 10696563. Dostupné z: doi:10.1111/acem.12876

[23] LIU, Vincent, Gabriel J. ESCOBAR, John D. GREENE, Jay SOULE, Alan WHIPPY, Derek C. ANGUS a Theodore J. IWASHYNA. Hospital Deaths in Patients With Sepsis From 2 Independent Cohorts. *JAMA* [online]. 2014, **312**(1) [cit. 2021-4-26]. ISSN 0098-7484. Dostupné z: doi:10.1001/jama.2014.5804

[24] REYNA, Matthew A., Christopher S. JOSEF, Russell JETER, Supreeth P. SHASHIKUMAR, M. Brandon WESTOVER, Shamim NEMATI, Gari D. CLIFFORD a Ashish SHARMA. Early Prediction of Sepsis From Clinical Data. *Critical Care Medicine* [online]. 2020, **48**(2), 210-217 [cit. 2021-4-26]. ISSN 0090-3493. Dostupné z: doi:10.1097/CCM.0000000000004145

[25] TRAN, Luan, Cyrus SHAHABI a Manh NGUYEN. *Representation Learning for Early Sepsis Prediction* [online]. In: . 2019-12-30, s. - [cit. 2021-5-5]. Dostupné z: doi:10.22489/CinC.2019.021

[26] ANDA DU, John, Nadi SADR a Philip DE CHAZAL. *Automated Prediction of Sepsis Onset Using Gradient Boosted Decision Trees* [online]. In: . 2019-12-30, s. - [cit. 2021-5-5]. Dostupné z: doi:10.22489/CinC.2019.423

[27] ZABIHI, Morteza, Serkan KIRANYAZ a Moncef GABBOUJ. *Sepsis Prediction in Intensive Care Unit Using Ensemble of XGboost Models* [online]. In: . 2019-12-30, s. - [cit. 2021-5-5]. Dostupné z: doi:10.22489/CinC.2019.238

[28] MORRILL, James, Andrey KORMILITZIN, Alejo NEVADO-HOLGADO, Sumanth SWAMINATHAN, Sam HOWISON a Terry LYONS. *The Signature-Based Model for Early Detection of Sepsis from Electronic Health Records in the Intensive Care Unit* [online]. In: . 2019-12-30, s. - [cit. 2021-5-5]. Dostupné z: doi:10.22489/CinC.2019.014

[29] MCCALL, John. Genetic algorithms for modelling and optimisation. *Journal of Computational and Applied Mathematics* [online]. 2005, **184**(1), 205-222 [cit. 2021-5-10]. ISSN 03770427. Dostupné z: doi:10.1016/j.cam.2004.07.034

[30] BAI, SHAOJIE, J. Zico KOLTER a Vladlen KOLTUN. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. [online]. 14 [cit. 2021-5-10]. Dostupné z: https://arxiv.org/pdf/1803.01271.pdf

Seznam symbolů, veličin a zkratek

XOR	Exkluzivní disjunkce
LSTM	Long short-term memory
TCN	Časová konvoluční síť – Temporal Convolutional Networks
ReLu	Rektifikovaná lineární funkce – Rectified Linear Unit
MB-GD	Mini - Batch Gradient Descent
ADAM	Adaptive Moment Estimation
MSE	Střední kvadratická odchylka – Mean Squared Error
MAE	Průměrná absolutní odchylka – Mean Absolute Error
CNN	Konvoluční neuronové sítě – Convolutional Neural Networks
RNN	Rekurentní neuronové sítě – Recurrent Neural Neworks
GRU	Gated Recurrent Unit
SOFA	Sequential Organ Failure Organ Assessment
JIP	Jednotka intenzivní péče
SVM	Metoda podpůrných vektorů – Support Vector Machine
SIRS	Syndrom systémové zánětové odpovědi – Systemic Inflammatory Response Syndrome
SAT	Střední arteriální tlak
SF	Srdeční frekvence
DF	Dechová frekvence
RS	Rozhodovací stromy
NaN	Not a Number
DSC	Dice koeficient
GA	Genetický algoritmus

A Obsah elektronické přílohy

- `/sidlo_david_BP_priloha` – kořenový adresář přiloženého souboru
  - `LSTM` – složka se sítí LSTM
    - `natrenovane_site` – složka s finálními sítěmi
      - `thres_LSTM3.mat` – prahová hodnota pro LSTM (3)
      - `thres_LSTM4.mat` – prahová hodnota pro LSTM (4)
      - `thres_LSTM5.mat` – prahová hodnota pro LSTM (5)
      - `thres_LSTM6.mat` – prahová hodnota pro LSTM (6)
      - `thres_LSTM6.mat` – prahová hodnota pro LSTM (7)
      - `trainedNet_LSTM3.mat` – natrénovaná síť LSTM (3)
      - `trainedNet_LSTM4.mat` – natrénovaná síť LSTM (4)
      - `trainedNet_LSTM5.mat` – natrénovaná síť LSTM (5)
      - `trainedNet_LSTM6.mat` – natrénovaná síť LSTM (6)
      - `trainedNet_LSTM7.mat` – natrénovaná síť LSTM (7)
    - `classWeights.mat` – hodnoty vah
    - `loss_graf.m` – skript sloužící k zobrazení grafu chybové funkce
    - `loss_LSTM.mat` – hodnoty chybové funkce
    - `main_LSTM.m` – hlavní skript sítě LSTM
    - `predikce_site_pacient.mat` – hodnoty sloužící k ukázce grafu predikce
    - `sepseLabel.mat` – anotace
    - `testovaci_data_pacient.mat` – hodnoty sloužící k ukázce grafu predikce
    - `weightedClassificationLayer.m` – skript váhové křížové entropie
  - `TCN` – složka se sítí TCN
    - `natrenovana_sit` – složka s finální sítí
      - `hyperparameters_TCN.mat` – hyperparametry pro TCN
      - `parameters_TCN.mat` – parametry pro TCN
      - `thres_TCN.mat` – prahová hodnota pro TCN
    - `initializeGaussian.m` – skript sloužící k inicializaci vah
    - `instanceNormalization.m` – skript sloužící k normalizaci
    - `leftPad.m` – skript sloužící k vyrovnání délky sekvence
    - `loss_graf.m` – skript sloužící k zobrazení grafu chybové funkce
    - `loss_TCN.mat` – hodnoty chybové funkce
    - `main_TCN.m` – hlavní skript sítě TCN
    - `maskedCrossEntropyLoss.m` – skript sloužící k učení sítě
    - `model.m` – skript sloužící k inicializaci modelu TCN
    - `modelGradients.m` – skript sloužící k učení sítě
    - `predikce_site_pacient.mat` – hodnoty sloužící k ukázce grafu predikce
    - `preprocessMiniBatch.m` – skript sloužící k inicializaci dávek
    - `residualBlock.m` – skript sloužící k inicializaci základního bloku
    - `spatialDropout.m` – skript sloužící k inicializaci dropout
    - `testovaci_data_pacient.mat` – hodnoty sloužící k ukázce grafu predikce
    - `thresholdL2Norm.m` – skript sloužící k L2 regularizaci
  - `compute_utility.m` – skript sloužící k výpočtu utility score
  - `compute_utility_for_ga.m` – skript sloužící k hledání prahové hodnoty
  - `dataset.m` – skript sloužící k načtení datasetu

- `nans.m` ..... skript sloužící k odstranění NaN hodnot z dat
- `predzpracovani.m` ..... skript sloužící k předzpracování
- `StandardizeDataCell.m` ..... skript sloužící k standardizaci dat
- `readme.txt` ..... návod pro spuštění